Simplifying UAS control and improving automation using an RBFNN for gesture and activity recognition

AP van Schalkwyk

orcid.org/0000-0002-3059-6824

Dissertation accepted in partial fulfilment of the requirements for the degree *Master of Science in Computer Science* at the North-West University

Supervisor:	Prof JV du Toit
Co-supervisor:	Mr H Foulds

Graduation May 2021

23489073

ACKNOWLEDGEMENTS

Above all else, guard your heart, for everything you do flows from it.

~ Proverbs 4:23

I would like to thank my Creator for providing me with the strength and wisdom to complete this study.

Thank you also to

My supervisor Prof Tiny du Toit for your support, guidance, patience and everything I was able to learn from you

My co-supervisor Mr Henry Foulds for all your ideas and contributions to this study

Dr Isabel Swart for proofreading and language editing

My parents Marietha and Adriaan van Schalkwyk for all your love and support. Thank you for being such wonderful parents

My aunts Joan Bekker and Brenda Kruger and uncles Frans Kruger and Piet Bekker for all your love and support.

Those who can imagine anything, can create the impossible.

~ Alan Turing

ABSTRACT

A simplistic view of an unmanned aircraft is that it is an aircraft with a computer system and a radio-link that replaces the pilot. The unmanned aircraft merely forms part of the total system referred to as an Unmanned Aircraft or Aerial System (UAS). UASs have become a pervasive technology in diverse fields, from recreational to advanced military applications. Similar to many modern technologies, UASs have become less expensive and simpler to attain. A UAS can be controlled manually or autonomously through various methods. Existing UAS control methods and the possible means to simplify UAS control and improve automation techniques are discussed. The first simplified control system uses accelerometer data from a smartwatch to control the pitch, yaw, and roll of the UAS. A gesture recognition system presented in this study was implemented by using a smartwatch that captures accelerometer data, based on the hand gestures made by the pilot. The accelerometer data is then analysed by a Fast Fourier Transformation (FFT) and sent to a radial basis function neural network (RBFNN) to identify the gestures. Various automated tasks are assigned to these identified gestures to improve automation and simplify UAS control. A following function is also introduced to enable the UAS to autonomously follow the pilot, based on the GPS coordinates of the hand-held device. The UAS is further automated by using an activity recognition system that uses an accelerometer to gather data from the smartwatch while the pilot is doing an activity. The data is processed by an FFT and used to train an RBFNN model which then predicts the activity that was performed. Based on the activity that was identified, the speed of the UAS has been adjusted accordingly. The experiments show that the RBFNN can accurately distinguish between the different gestures. The RBFNN is also able to accurately predict the activity that was performed by the pilot. UAS control is simplified by enabling the UAS to autonomously follow the pilot, based on GPS coordinates, enabling the pilot to focus on other activities instead of flying the UAS. The ability of the RBFNN to identify gestures performed by the pilot enables the pilot to easily send control instructions to the UAS without the need to carry a bulky remote control in his/her hand. The RBFNN also improves UAS automation by autonomously identifying the activity the pilot is performing and then adjusting the speed accordingly, allowing the pilot to focus on other activities.

Key terms: Artificial Intelligence Agents, Automation, Fast Fourier Transformation, Radial basis function neural network, Unmanned aerial vehicle, Unmanned Aircraft or Aerial System, UAS Control methods.

OPSOMMING

’n Vereenvoudigde siening van 'n onbemande vliegtuig is dat dit 'n vliegtuig is met ’n rekenaarstelsel en radioskakel wat die vlieënier vervang. Die onbemande vliegtuig vorm slegs 'n deel van die algehele stelsel wat bekend staan as 'n onbemande vliegtuig of lugstelsel (OLS). OLS’e het 'n deurdringende tegnologie op verskillende terreine geword, van ontspannings- tot gevorderde militêre toepassings. Net soos baie moderne tegnologieë, het OLS’e goedkoper en eenvoudiger geword. ’n OLS kan deur middel van verskillende metodes met die hand of outomaties beheer word. Bestaande OLS-beheermetodes en die moontlike maniere om OLS-beheer te vereenvoudig, asook die verbetering van outomatiseringstegnieke word bespreek. Die eerste vereenvoudigde beheerstelsel gebruik versnellingsmeterdata van 'n slimhorlosie om die toonhoogte, draai en rol van die OLS te beheer. Die gebaarherkenningstelsel wat in hierdie studie bespreek word, is geïmplementeer deur gebruik te maak van 'n slimhorlosie wat versnellingsmeterdata vaslê wat gebaseer is op die handgebare wat deur die vlieënier gemaak word. Die versnellingsmeterdata word dan ontleed met ’n Vinnige Fourier Transformasie (VFT) en na 'n radiale basisfunksie neurale netwerk (RBFNN) gestuur om die gebare te identifiseer. Verskeie outomatiseringstake word aan hierdie geïdentifiseerde gebare toegewys om die OLS-beheer te verbeter en outomatisering te vereenvoudig. Daar word ook ’n volgfunksie ingestel om die OLS in staat te stel om die vlieënier outonoom te volg op grond van die GPS-koördinate van die handtoestel. Die OLS word verder geoutomatiseer deur gebruik te maak van 'n aktiwiteitherkenningstelsel wat ’n versnellingsmeter gebruik om data vanaf die slimhorlosie in te samel, terwyl die vlieënier 'n aktiwiteit doen. Die data word verwerk deur ’n VFT en gebruik om 'n RBFNN-model op te lei met die aktiwiteit wat uitgevoer is en om die aktiwiteit dan te voorspel. Op grond van die aktiwiteit wat geïdentifiseer is, word die spoed van die OLS dienooreenkomstig aangepas. Die eksperimente wys dat die RBFNN akkuraat tussen die verskillende gebare kan onderskei. Die RBFNN is ook in staat om die aktiwiteit wat deur die vlieënier uitgevoer is, akkuraat te voorspel. OLS-beheer word vereenvoudig deur die OLS in staat te stel om die vlieënier outonoom te volg op grond van die GPS-koördinate. Dit stel die vlieënier in staat om op ander aktiwiteite te fokus in plaas daarvan om die OLS te beheer. Die vermoë van die RBFNN om gebare wat deur die vlieënier uitgevoer word, te identifiseer, stel die vlieënier in staat om beheerinstruksies maklik aan die OLS te stuur sonder om 'n lywige afstandbeheer in sy/haar hand te dra. Die RBFNN verbeter ook OLS-outomatisering deur die aktiwiteit wat die vlieënier uitvoer outomaties te identifiseer en dan die spoed daarvolgens aan te pas, sodat die vlieënier op ander aktiwiteite kan fokus.

Sleutelterme: Kunsmatige Intelligensie-agente, Onbemande vliegtuig of lugstelsel, Outomatisering, Radiale basis funksie neurale netwerk, OLS-beheermetodes**,** Onbemande lugvaartuig, Vinnige Fourier Transformasie.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

A simplistic view of an unmanned aircraft[1] is that it is an aircraft with no pilot on-board (Austin, 2010). A computer system and a radio-link replace the pilot, but it is more complicated than that. The unmanned aircraft merely forms part of the total system referred to as an Unmanned Aircraft or Aerial System (UAS). The UAS has become a pervasive technology in diverse fields, from recreational to advanced military applications (Haluza & Cechak, 2016). Like many modern technologies, UASs have become less expensive and simpler to attain (Haluza & Cechak, 2016).

In this chapter, a background on UASs is given in Section 1.1. Defining automation is discussed in Section 1.2. Key UAS terminology is presented in Section 1.3, followed by a discussion on UAS applications in Section 1.4. In Section 1.5, UAS control automation is discussed. The problem statement is given in Section 1.6, followed by the research aim and objectives in Section 1.7. Ethical considerations are discussed in Section 1.8. The outline of the chapters is considered in Section 1.9, and in Section 1.10, the conclusion to the chapter is given.

1.1 Background

Before beginning the review of the present research and discussion of the aim of this study, it is beneficial to investigate and understand the history of UAS technology. The history of unmanned aircraft is the history of all aviation (Leylek & Costello, 2012). From centuries past, when the Chinese kites graced the skies to the first hot air balloon, unmanned flying was investigated first before the risk of testing manned aircraft. One of the earliest applications of the unmanned flight was by the Chinese general, Zhuge Liang. Liang used paper balloons fitted with oil-burning lamps that heated the air inside the balloon. Liang flew the balloons over the enemy at night to let them believe there was a divine force at work. Next, other forms of vertical flight and history are discussed.

1.1.1 Kites

A kite has a thin piece of string attached to it, and this is used to control the kite by pulling on the other end of the string. This is a form of unmanned aerial control in the same sense that a UAS is controlled by a controller that is held in a person’s hand.

The exact origin of the kite is not known, but there are a few possible theories on the origination. One of these theories, as suggested by Waley (1936), is from an ancient Chinese method

[1] Although some people may find the term *unmanned* dated or non-inclusive in this context, it is frequently used in literature on Unmanned Aircraft or Aerial Systems.

where an arrow is shot with a line attached to it. When the arrow is shot, it could then be retrieved by hauling it in.

One of the earliest forms of an unmanned application was when Chinese general Han Hsin flew a kite over the city walls of his enemy. Hsin wanted to measure the distance required by his army to dig a tunnel that would go past the enemy's defences (Needham, 1965). During World War I, man-lifting kites were invented for enemy observation (American Kitefliers Association, 2016). Today, a UAS can be used, and there is no need for a pilot to be on board.

1.1.2 Hot-air balloons

Hot-air balloons are one of the oldest successful human flight technologies (Federal Aviation Administration, 2012). On 19 September 1783, Pilatre De Rozier launched the first hot-air balloon called “Aerostat Reveillon” into the air (Walker, 1975). The first passengers were a sheep, a duck, and a rooster. The first human-crewed hot-air balloon flight happened about two months later in Paris and flew for about 20 minutes. This was one of the first manned vertical flights. Unmanned air balloons equipped with sensors are currently used by the National Weather Service to predict the weather. Hot-air balloons are also found in the history of the military. Napoleon used anchored observation balloons in some of the battles and considered using balloons to ferry troops in his proposed invasion of England (Federal Aviation Administration, 2012). Today, UASs are used for military observations, and it is easier to control a UAS than it is to control a balloon.

1.1.3 Unmanned aerial vehicles and systems

Unmanned aerial vehicles (UAV) have captivated the imagination of humans for centuries (Valavanis & Vachtsevanos, 2015). The idea of a flying vehicle was thought to have first been perceived about 2 500 years ago, in ancient Greece and China. Pythagoras, Archimedes, and other inventors studied the use of autonomous machines for various applications. Archytas, from Tarentum in the south of Italy, is known as the Leonardo da Vinci of the ancient world. In 425 B.C. he developed the first autonomous flying machine, a mechanical bird made of wood, known as the *pigeon*. The bird flew using compressed air that was in the form of steam and was enclosed in the bird's stomach. The bird flew for about 200m before it started to fall to the ground after its energy was depleted. During the same era, in China, the first concept of an aircraft that could achieve vertical take-off and landing (VTOL) was documented (Leishman, 2006). VTOL is a technique in which the aircraft rises directly into the air and lands vertically onto the ground. The device consisted of feathers at the end of a stick that was spun between hands to generate enough lift before it was released into the air for free flight. In 1483 Leonardo da Vinci designed an aircraft called the *aerial screw* (Valavanis *et al*., 2007). He envisioned that

the device could be capable of hovering. From these examples, it is apparent that man has been fascinated by flight for ages.

UAVs, unmanned ground vehicles (UGVs), unmanned surface vehicles (USVs) (operating on water surfaces), and unmanned underwater vehicles (UUVs) all share a few standard features. Some of these features include architecture, propulsion, communication (which includes control), and technology that aims to improve autonomy (Valavanis & Vachtsevanos, 2015). These types of vehicle can perform various tasks in the fields of law enforcement, environmental monitoring, disaster relief, and recovery operations. The payload types of these vehicles are also significantly different, depending on sensing requirements, mission objectives, and various other factors. The military community primarily drives the most significant growth experienced in the UAV design and development sector. However, the growth is expected to gravitate more towards the civilian and public domains in the future.

In the early years of aviation, the idea of having an aircraft with no pilot on board had the advantage of removing the risks to the pilot who was involved with manned flights (Leylek & Costello, 2012). The growth in the UAS development area has increased significantly due to the development of lightweight construction materials, microelectronics, signal processing equipment and GPS navigation (Finn & Wright, 2012).

An unmanned system (US) is defined as an air, maritime, or ground vehicle that has no pilot on-board (Valavanis & Vachtsevanos, 2015). An unmanned system can be manually controlled, or it can fly autonomously with the help of pre-programmed flight plans (Rouse, 2015). The US can be expendable or recoverable and can carry a variety of payloads similar to a UAV. The payloads of a US are usually determined by the type, functionality, operational characteristics, and mission objectives of the US (Valavanis & Vachtsevanos, 2015).

As seen above, unmanned flights have a rich history that dates back to ancient times. The first systems that can be associated with the modern definition of UAV are recent, and mainly include reconnaissance drones that were developed during the Cold War. The design and development of modern UAVs have expanded and evolved into different architectures, for example, quadrotors, ducted fan, and blimps in addition to the classical helicopter and fixed-wing approaches. In modern times, the unmanned aircraft has become a more autonomous air vehicle that flies to mimic and improve on the manoeuvres of manned aircraft. There are many benefits to the use of unmanned aircraft, but interacting with the environment can be challenging (Becerra, 2019). It is beneficial to investigate automation to determine how it can be incorporated with UAS control to limit these challenges (some of these challenges are discussed in Section 1.5). Next, a brief introduction to automation is given. A more detailed review of automation is provided in Chapter 3.

1.2 Defining automation

Automation is defined as a machine agent which is capable of carrying out functions that are usually performed by a human (Vincenzi *et al*., 2015). The definitive aim of automation is to replace human control, planning and problem solving with autonomous devices and computers (Bainbridge, 1983). It is often believed that the process of replacing manual operations with fully automated operations can be done by simply replacing the manual operator with a robot or advanced machine (Lindstr *et al.*, 2008). In most cases, this is not the truth. Automation can be the full or partial replacement of functions carried out by a human operator (Parasuraman *et al.*, 2000). The advantage is that it can reduce the risk and cost in terms of time and money (Rafi *et al.*, 2006). Implementations of automation can vary, and this introduces a broad spectrum of automation levels. Levels of automation, as defined by Parasuraman *et al.* (2000), rank from level one as the lowest level of automation to level ten the highest level of automation. These levels are as follows:

- **Level 1:** The computer does not assist, and the human is left to make all the decisions and take action.
- **Level 2:** The computer gives a complete set of decisions, actions, and alternatives.
- **Level 3:** The computer narrows the selections down to a few.
- **Level 4:** The computer suggests one alternative.
- **Level 5:** The computer executes a suggestion only if the human approves.
- **Level 6:** Allows the human a certain amount of time to veto before performing an autonomous action.
- **Level 7:** Does autonomous actions and then informs the human of the actions.
- **Level 8:** Provides information to the human only if required.
- **Level 9:** Informs the human only if the computer decides it needs to.
- **Level 10:** The computer does everything autonomously and ignores the human.

Automation, as applied to a UAS, refers to the ability of a UAS to complete a mission with minimal human interactions involved (Vincenzi *et al*., 2015). The goal of UAS automation is to reduce the need for human-to-UAS interaction in such a way that fewer human operations are needed to fly the UAS (Rafi *et al.*, 2006). In reality, an UAS will exhibit at least some degree of automation (Clarke, 2014). The stabilisation of attitude (the orientation of an aircraft with respect to the horizon) and altitude is an example of automation that is needed on most UASs. It is arguably better to perform these functions autonomously than to rely on a human pilot to be in control of these tedious, repetitive tasks. Other higher-level functions, such as take-off and

landing are often delegated from a human pilot to an auto-pilot function. It might be prudent to discuss modern terminology regarding UAS technology before continuing the discussion of the technology itself. The next section contains an overview of the different terminologies that are used to describe unmanned aircraft.

1.3 UAS terminology

Various terms and acronyms are used in the field of unmanned aerial vehicles, more commonly referred to as drones. These terms and acronyms are included to provide basic definitions and clarify some confusion and misconceptions that might exist. An overwhelming number of terms is used that describes the same concept. The different terms are often a result of the diverse requirements and concepts between military and civilian uses of these unmanned aircraft. Regulation and legal considerations also play a role in the definition of these terms. Several names have been given to describe unmanned aircraft. The American Federal Aviation Administration (FAA) uses the term Unmanned Aircraft System or Unmanned Aerial System (UAS) (Federal Aviation Administration, 2008). Remotely Piloted Vehicles (RPVs) is another term that was used in the Vietnam War. Today, the United States Air Force Unmanned Aircraft Systems Flight Plan (USAFP) has substituted Remotely Operated Vehicle (ROV) for Remotely Piloted Aircraft or RPA. This term is used to include the aircraft and the pilot. The United Kingdom has chosen Remotely Piloted Aircraft System (RPAS) as the preferred term to demonstrate that a person controls them (Federal Aviation Administration, 2008).

Unmanned Aerial Vehicles (UAVs), also known as drones, refers to aircraft with no pilot on board (Joint Capability Group on Unmanned Aerial Vehicles, 2010). The word “unmanned” implies that there is no human who actively pilots the aircraft onboard. The unmanned aircraft is controlled by an onboard automated system or with a remote control. Various definitions have been given to describe the term Unmanned Aerial Systems. For several years, the term UAV was used, and the Joint Capability Group on Unmanned Aerial Vehicles defined an Unmanned Aerial System as a reusable aircraft that is developed to do operations without an on-board pilot. The aircraft does not carry any passengers and can be piloted remotely or with the use of pre-programmed flight plans to fly autonomously. In the description above, the word reusable is characterised to differentiate between an unmanned aircraft and a guided weapon or munition delivery system.

The U.S. Department of Defence, U.S. Federal Aviation Administration, and the European Aviation Safety Agency adopted the same UAS term. These agencies consider a UAS an aircraft which can have external systems that consist of ground control stations, communication links, and launch and retrieval systems in addition to the aircraft itself (U.S. Department of

Defense, 2000). The FAA defines an Unmanned Aircraft as a device that is used or is intended to be used for flight in the air without a pilot onboard (Federal Aviation Administration, 2008). It includes all classes of aeroplane, helicopter, and airship that have no pilot onboard. Unmanned aircraft include three axes that can be controlled, thus excluding traditional balloons.

The European Aviation Safety Agency (EASA) defines a UAS as an Unmanned Aircraft System with an individual system element that includes an unmanned aircraft, a control station, and any other system element that is necessary to enable flight with a command and control line and a launch and recovery element (Morier, 2010). In practice the term UAV and UAS are often similar and only when the system aspect is essential, does the term UAS have a preference.

In this study, the term UAS will be used to refer to the system and the term UAV to refer to the aircraft itself. The term UAS encompasses all the aspects of deploying these aircraft and not just the platform itself (Marshall, 2009). As mentioned before, various types of UAS can be used to perform various functions. Some of the possible UAS applications are discussed in the next section. These applications will provide insight into the technologies involved, the control of the system and the possibilities of automation.

1.4 UAS applications

The field of unmanned aircraft systems has grown exponentially over the past 20 years with military applications dominating the field. In 2013, a turning point was reached, and applications moved more to the commercial and public domain for the following three crucial reasons (Valavanis & Vachtsevanos, 2014):

- The European Remotely Piloted Aircraft Systems (RPAS) Steering Group handed over to the European commission the "*Roadmap for the Integration of Civil Remotely Piloted Aircraft Systems into the European Aviation System*" (European RPAS Steering Group, 2013). This roadmap aims at facilitating decisions taken by the involved organisations, providing transparency and efficiency in planning different initiatives, and supporting the coordination of related activities in Europe.
- In November 2013, the U.S. Federal Aviation Administration (FAA) presented the "*Roadmap for Integration of Civil Unmanned Aircraft Systems (UAS) in the National Airspace System (NAS)*" (Huerta, 2013), which was created by the FAA and UAS Aviation Rulemaking Committee (ARC). The roadmap outlined the tasks and considerations that need to be completed to enable UAS integration into the NAS.
- In December 2013, the FAA announced the six UAS research and test site operations across the United States that would be studying the broad range of issues and challenges related to the use of unmanned aircraft. The FAA also stated that they are

fully committed to the safe and efficient integration of UASs into the NAS to enable this emerging technology to be fully utilised.

The actions mentioned by the European RPAS steering group and the FAA have opened the door to utilise unmanned aircraft in the civilian and public domain. The most prominent applications from literature are briefly mentioned in this section to illustrate the pervasive role UASs play in the civilian and public domain.

1.4.1 Remote sensing

Remote sensing is the technology used to obtain information about objects or certain areas from a distance by using an aircraft or satellite (Dyring, 1973). The use of a UAS for remote sensing applications has increased significantly over the past few years (Stark & Chen, 2016). Traditionally, satellites are used for remote sensing (Bhardwaj *et al.*, 2016). Satellite remote sensing is useful, but is limited by the amount of data that is collected, and the cost of data acquisition is also high. Remote sensing with the use of a UAS can be more cost-effective, and minimal effort is required (Holness *et al.*, 2016). A UAS is easy to deploy, and the sensors can easily be altered (Bhardwaj *et al.*, 2016). An example of a sensor that is used on a UAS for remote sensing is an optical camera. It is also the core component that is used in any remote sensing system (Toth & Jóźków, 2016). A UAS can also be automated for optimal land coverage remote sensing (Bhardwaj *et al.*, 2016).

1.4.2 Environmental monitoring

Over the past few years, there has been an increasing interest in developing tools that can aid with environmental monitoring (Ventura *et al.*, 2016). In marine ecology, for example, it is essential to gather data about the seabed, landform, and other topographic data to define and map critical marine habitats. It has often been challenging and time-consuming to get access to this kind of data with a high level of detail for shallow and inaccessible marine habitats. With the development of new technologies, it is possible to overcome these constraints. UASs have been developed that can produce a highly accurate aerial map of the fish in a nursery area. The technology is inexpensive and straightforward to use, and it can produce aerial photographs of the marine areas. With environmental monitoring spatial analysis of imagery and geographic information systems, data import is sometimes required (Perry *et al.*, 2008). A simultaneous collection of imagery with positional data is needed to acquire georeferenced images. UASs have positioning capabilities within the autopilot system, but to achieve a higher level of accuracy, it is essential to also use ground control points (GCPs). Although this solution provides a higher level of accuracy, it is not always a trivial solution, especially in impervious coastal areas. Research has shown that it is possible to eliminate GCPs by using a low-altitude, long-endurance UAS to collect precise data for direct image georeferencing.

1.4.3 Aerial surveillance

An aerial surveillance system is an aerial vehicle that can be controlled remotely with capabilities to transmit real-time data to a ground control station (Zaheer *et al.*, 2016). There is a growing need for surveillance in today's world to ensure the safety and security of people. Several civilian and military applications require aerial surveillance. Some civilian aerial surveillance applications include the monitoring of forest fires, oil fields and pipelines, and the tracking of wildlife (Beard *et al.*, 2006). Numerous military applications make use of aerial surveillance. The traditional approach to these applications is the use of manned vehicles for surveillance. Manned vehicles are expensive and mostly large. These manned vehicles are operated in hazardous environments and can be life-threatening to the pilot. The development of automated aerial surveillance systems with the help of UASs makes it easier to address these applications. Various military aerial surveillance missions, such as the surveillance of unknown areas, forest conservation, and spying on enemy territory can be performed with the use of a UAS (Ma'Sum *et al.*, 2013). A UAS has highly efficient electric motors that allow for discrete and almost silent aerial surveillance. Police officers can also use these systems to patrol within a city to ensure that citizens are safe and that law and order are maintained (Zaheer *et al.*, 2016).

Natural disasters, such floods can cause a large number of casualties and material damage (Popescu *et al.*, 2015). UASs have been successfully used for flood area detection. There is a need to examine these areas that were affected before a search and rescue mission can be attempted (Zaheer *et al.*, 2016). UASs can be used to complete these tasks faster, and natural obstacles, such as steep mountain slopes, dangerous water currents, hostile desert areas and other dangerous areas can easily be overcome with the use of a UAS. Images or video streams are taken from UASs that cover large areas of observation. Contrary to this, satellites and fixed cameras are much more expensive to use than a UAS (Popescu *et al.*, 2015).

Autonomous UAS aerial surveillance is also possible. Nag *et al.* (2017) used Automatic Dependent Surveillance (ADS) transponders to automate aerial surveillance. The ADS transponders of the aircraft provide surveillance data. This data includes aircraft position, velocity, navigational intent and meteorological data and is automatically transmitted without operator input.

1.4.4 Agriculture

The continuous innovation and progression in UAS technology have facilitated a series of applications in the field of agriculture (Katsigiannis *et al.*, 2016). Satellites with optical and multispectral technologies are used to capture images, which are processed to assist with the

correct production and usage of resources (Tripicchio *et al.*, 2015). These techniques help to assess the state of healthy farming. The quality of these techniques with the use of satellites depends on the height and density of the crop, and it can also produce low-quality data when visibility is influenced by, for example, clouds (Fernandez *et al.,* 2016). This method is difficult and costly (Grenzdörffer *et al.*, 2008). A more cost-effective alternative is to capture low altitude, high-resolution images using a UAS. This approach has the added benefit of being less susceptible to the influences of cloud cover, although the coverage area of this approach can be a drawback (Fernandez *et al.,* 2016). The UAS is equipped with a GPS and digital camera to enable it to fly autonomously and to navigate in the area of interest (Grenzdörffer *et al.*, 2008). Aerial photography produced by a UAS has several advantages for agriculture (Chivasa *et al.*, 2017). A UAS can be quickly and repeatedly deployed, is cost-effective, safer than piloted aircraft, flexible in terms of the height and duration of the flight and the images that are obtained are of high resolution. The images that are gathered can be used to observe individual plants, patches, gaps, and patterns over the fields. Farmers use a UAS to map an entire farm, scout for weeds and pests, and spot dry areas or plants with diseases.

Several researchers have developed UASs for the agricultural sector. Herwitz *et al.* (2004) developed a UAS which uses solar power to collect high-spatial, multispectral images of a coffee plantation. The images were used to map the ripeness of the coffee in a plantation and to identify drip irrigation and weed problems. Hunt *et al.* (2006) used a UAS with five standard digital cameras and modified the filters of the camera lenses to acquire near-infrared images. This system can be used to monitor crop nitrogen requirements.

The use of protected materials is a crucial component for pest management in agriculture. Fertilisers and chemicals are also frequently needed at specific times and locations to manage crops (Huang *et al.*, 2009). These applications are generally made with the use of ground sprayers or aeroplanes. These methods are well suited for large cropping areas, but they may become inefficient or cumbersome when the application is needed on a smaller field. UASs that are cheaper and easier to operate may serve to address this need. A spray system was developed and successfully mounted on a UAS. The spray system directly communicated with the UAS electronic system that enabled the spray to be released, based on specified GPS coordinates and pre-programmed spray locations (Huang *et al.*, 2009). UASs can apply chemicals much closer than traditional methods (the use of a manned aircraft) (Stark & Chen, 2016). This means that there is less potential of the chemicals drifting to unwanted areas. The disadvantage is that the UAS requires an increased payload. Stark *et al.* (2016) proposed the use of a network of small UASs that autonomously spray the chemicals to reduce the payload.

1.4.5 Search and rescue

When a search and rescue attempt is made, every second counts (Güldenring *et al*., 2020). Any delay can result in dramatic consequences or even human losses (Waharte & Trigoni, 2010). It is also essential to have a rapid overview of the situation, and the view is often only possible from the sky (Güldenring *et al*., 2020). It takes helicopters and planes a certain amount of time to be ready for deployment, while a UAS can be put into action almost immediately. UASs have been developed with a carbon fibre housing which allows them to fly in conditions with snow, rain, and extreme weather (Klimkowska *et al.*, 2016). Data transfer occurs in real-time between the UAS and the ground station, and it is therefore no longer necessary for search and rescue personnel to enter dangerous areas to analyse the situation. These UASs are also equipped with various sensors that identify the amount of gas and smoke composition of fires. The information is used by the rescue personnel to inform the firefighters of the dangers. UASs have been developed by a group of Swiss researchers that can navigate forest trails autonomously to find people and aid with search and rescue attempts (Giusti *et al.*, 2015).

In a typical search and rescue scenario, the UAS is deployed in the area of interest equipped with sensors. The sensors on the UAS are used for finding evidence of possible victims. The finding is then communicated to a remote ground station or the search and rescue team (Waharte & Trigoni, 2010). In 2006, two UASs were deployed to survey the damaged area and search for possible survivors after Hurricane Katrina.

A review of some of the UAS applications was presented in this section. It is evident that there has been a shift in the field of unmanned aircraft systems and the focus is no longer just on military applications. From the discussion in this section, a variety of control methods has been identified. Most of the control techniques are either remote-controlled or make use of a ground control station. Researchers are also investigating automation techniques for various functions to improve UAS control. In the next section, a brief introduction to UAS control automation is given.

1.5 UAS control automation

UAS control technologies have improved over the past few years (Chen *et al.*, 2009). A UAS can perform increasingly complex autonomous manoeuvres, but most UASs are not fully autonomous and are mostly operated remotely by humans (Becerra, 2019). Human remote piloting may not always be sufficient due to signal strength or in the case where speed and precision, which may be outside human capabilities, are needed to control the UAS. The process of fully automating UASs requires research on various technologies and algorithms. For example, UAS obstacle avoidance needs to be improved. Obstacle avoidance becomes

particularly important, as autonomous UASs start to share civil airspace that is also used by other aircraft. A detailed discussion of UAS control automation is given in Chapter 3. After introducing UAS technologies and some of the possible applications, the problem statement is given in the next section.

1.6 Problem statement

There are UASs that are capable of performing complex autonomous tasks, such as obstacle avoidance, landing on the ground or docking into a station. However, most UASs are not fully autonomous (Becerra, 2019). Humans remotely pilot most UASs. Human remote piloting might not always be possible; for example, there might be a problem with the data communication link or the task that needs to be performed might be outside of the pilot's capabilities. To improve UAS automation, many technological and algorithmic advances are still needed (Becerra, 2019).

For many UASs, the problem is for a person to manually control the UAS while also performing an activity (Mugnier, 2015). Most UASs are remotely piloted with a remote control or a smartphone (Anand & Mathiyazaghan, 2016). These remote controls and smartphones can be rather bulky and need to be in the pilot's hand to control the UAS effectively. It can be rather difficult for the pilot to have the remote control in his/her hand while performing activities, such as running, cycling, and climbing. It is, therefore, advisable to investigate methods to simplify and automate UAS control.

From the applications mentioned in Section 1.4, it can be seen that some of these applications can benefit from autonomous target tracking. The possibility of having a UAS follow a person may have a tremendous impact on, for example, search and rescue scenarios (Giusti *et al.*, 2015). Autonomous target tracking also removes the responsibility of the pilot to manually and continuously control the UAS. This enables the pilot to focus on other tasks, and it also makes it easier (less skill is required to control the UAS) to deploy the UAS. There are currently UAS implementations that can follow people while they are doing different kinds of activity, for example, the DJI spark (DJI, 2017). Having the UAS autonomously follow the pilot already removes much responsibility from the pilot. It is also helpful in enabling the pilot to still be able to send control instructions to the UAS. The control needs to be simple, intuitive, and non-intrusive, without the need for a sophisticated, bulky remote control device.

There has been a significant increase in the popularity of smartwatches over the past few years (Xu *et al.*, 2015). The increase in the popularity of smartwatches presents a unique opportunity. Since the smartwatch is worn on the wrist, it can be used to understand the user's hand and

arm movements. Most smartwatches have a built-in accelerometer and gyroscope sensor. The sensor data can be recorded and analysed to identify the user's hand and arm movements.

The research question investigated in this study is: "Can hand gestures and activities that are classified by a Radial basis function neural network (RBFNN) be used to simplify UAS control and improve UAS automation?" To address this research question, the aim and objectives of the study are discussed in the next section.

1.7 Research aim and objectives

The primary aim of this study is to simplify UAS control and improve automation, using an RBFNN for hand gesture and activity recognition.

To achieve the primary aim, the following secondary objectives must be met:

1. Perform a literature review on UAVs, UASs, applications of UASs, the positivistic research paradigm, the design science research strategy, the SCRUM development methodology, control systems (manual and autonomous), automation of UAS control techniques, artificial neural networks, RBFNNs and the FFT algorithm.
2. Implement a basic UAS control system which will act as the starting point for comparisons.
3. Improve the system in (2) by controlling the UAS with smartwatch accelerometer data.
4. Enhance the system in (3) with gesture recognition implemented through an RBFNN.
5. Extend system (4) by implementing a pilot follow function to simplify UAS control.
6. Improve UAS control of (5) with automated activity recognition, using an RBFNN.
7. Evaluate the implemented systems, based on the simplification of control and the improved levels of UAS automation.

In the following section, there is a brief discussion of the ethical considerations for the study.

1.8 Ethical considerations

The research proposal was presented to the ethics committee of the Faculty of Natural and Agricultural Sciences for ethical clearance. The study was approved as a no-risk study and the ethics number: NWU-01207-19-A9 was issued on 10 October 2019.

1.9 Outline of chapters

In this section, a brief discussion of each chapter of the dissertation is provided.

Chapter 1 – Introduction: An introduction to the dissertation that provides a general background on UASs, applications and key terminology. The problem statement, research aim, secondary objectives and dissertation outline are also discussed.

Chapter 2 – Research method: A description of the research paradigm, methods and design methodologies used for the research in this study are presented.

Chapter 3 - Literature review: Previous research related to UAVs, control systems, and automating UAS control is discussed. Neural networks in general and the radial basis function neural network used to perform gesture and activity recognition is considered. The Fast Fourier Transformation utilised for data pre-processing is examined.

Chapter 4 - System design and implementation: A discussion on how the various systems were implemented to simplify UAS control and improve automation is given.

Chapter 5 - Results and discussion: The results from the systems implemented are documented and examined.

Chapter 6 - Conclusion: In this chapter, the aim and secondary objectives of the study are revisited to determine if they were met. A summary of the contributions is presented. Finally, possible future work is suggested.

The following section contains a conclusion to this chapter.

1.10 Conclusion

UASs have become a pervasive technology in diverse fields, from recreation to advanced military applications. UASs are now an extension of the human desire to create innovative solutions and serve as entertainment. Like many modern technologies, UASs have become less expensive and easier to attain. The field of unmanned aircraft systems has grown exponentially over the past 20 years with military applications dominating the field (Valavanis, K.P. & Vachtsevanos, 2014). There are different unmanned aircraft available, as discussed at the beginning of this chapter. The use of a UAV has become the preferred unmanned aircraft for various applications. With each application that was mentioned, it is evident that researchers are investigating techniques to simplify the process of completing tasks with a UAS. Automation plays a vital role in improving and simplifying these tasks. Automation techniques are also used for addressing various challenges with manual UAS control, as mentioned in Sections 1.5 and 1.6. Information about the background, scope and objectives of the research were provided. The research methodology that will be followed in this study is considered next.

CHAPTER 2 RESEARCH METHOD

In this chapter, a brief review is given of the research methodology and aspects that are adopted for the study. The research of this study focuses on scientific research that involves the use of a systematic process, being objective, and obtaining a multitude of information that is used for analyses in order to draw a conclusion (Blankenship, 2010). The process focuses on systematically testing several ideas in such a manner that another individual could conduct the same study again. In this chapter, Section 2.1 is a discussion on the research paradigm, followed by a discussion on the research strategy in Section 2.2. The development methodology is discussed in Section 2.3. In Section 2.4, a conclusion to the chapter is presented.

2.1 Research paradigm

A paradigm is a set of mutual presumptions or means of thinking about some aspects of the world (Oates, 2015). Different communities share different ways of researching to create and obtain knowledge. There are several research strategies used in information systems research, each with its underlying philosophical paradigm. The positivist paradigm assumes an objective reality that is used by researchers to compare their claims and determine the truth. In this section, the positivistic paradigm is discussed.

2.1.1 The positivistic paradigm

Positivism, interpretivism, and critical research are three of the main philosophical paradigms. Of these three paradigms, positivism is the oldest (Oates, 2015). Positivism is known as the scientific method, and the approach used in natural science research. The positivistic paradigm is well established and has evolved over the past 500 years from the time of Bacon, Galileo, and Newton. The term positivistic research can be used to refer to all the approaches in science where it is assumed that scientific knowledge can only be gathered through data that are directly observed (Susman & Evered, 2016). A positivistic paradigm is used in this study. The results of the experiments in the study need to be obtained through quantifiable observations, must be repeatable, and yield similar results when the experiment is repeated. In the next section, the ontology of positivism is discussed.

2.1.1.1 Ontology

In general, ontology is the study or concern of the types of things that exist and what objects there are in the universe. The term ontology is derived from the Greek term *onto* which means being and *logia* that translates to written. In information technology, ontology is defined as the working model of entities and interactions which are in some domain of knowledge or practices

(Larose & Kruse, 2005). The positivist ontology believes that the world is external and that there is only one objective reality to any research phenomenon or condition regardless of what the researcher's perspective or belief is (Edirisinga, 2012). Positivistic ontology takes a controlled and structural approach to research by identifying a precise research topic and constructing an appropriate hypothesis. A positivist believes that reality is objectively given and can be measured using properties which are not dependent on the researcher and his or her instruments. In other words, a positivist believes that knowledge is objective and quantifiable. The positivistic researcher adopts scientific methods and knowledge acquisition with the help of quantification to enhance the accuracy of describing the parameters and relationships among them. Positivism is concerned with discovering the truth and presenting it through empirical means. In the next section, the epistemology of positivism is discussed.

2.1.1.2 Epistemology

Epistemology is the study of knowledge. Epistemologists are concerned with the nature of knowledge and the degree of human knowledge. The nature of knowledge is what it means to say that someone knows or fails to know something. The extent of knowledge is how much we know or how much we can know (Truncellito, 2015). The positivist epistemology accepts that facts can only be derived from the scientific method that can make valid knowledge claims. Positivism assumes that the researcher is separate from the research and does not affect the outcome of the research (Hjorland, 2009). The results that are gathered should be data-driven and should not be influenced by what the researcher believes. Next, the characteristics of positivism are discussed.

2.1.1.3 Characteristics

Characteristics are the features or qualities associated with a person, place, or thing and assist in identifying them. The characteristics of positivism are as follows (Priya, 2015):

- Science is the only valid source of knowledge.
- Facts are the entity of knowledge.
- The philosophy does not retain a method different from science.
- Positivism denies intuition, prior reasoning, and theological and metaphysical knowledge.
- All knowledge derived through science must be based on direct experience.

Other characteristics of positivism include the following (Watt, 2010):

- Positivism believes that the laws of the universe govern every phenomenon.
- It claims that once the knowledge is gained, it can be used to explain events.

- Positivist research is structured around the ideologies of verifiability or falsifiability.

Positivistic characteristics, according to Oates (2015), are the following:

- **The world exists independently of humans:** There is a social and physical world that exists which is not just in the human mind that can be studied, captured and measured. For example, the law of gravity will still exist even if some natural disaster or human accident wiped out the human population.
- **Measurement and modelling:** The researcher discloses the world by making observations, measurements and creating models of how the world works.
- **Objectivity:** The researcher is neutral and objective and seen as an impartial observer. Facts about the world can be discovered independently of what the researcher's values and beliefs are.
- **Quantitative data analysis:** This method of data analysis has a strong preference for mathematical modelling, proofs and statistical analysis. With the use of mathematics, the researcher can give a logical, objective means of analysing observations and results.
- **Universal laws:** The researcher seeks generalisations: universal laws, patterns or indisputable facts that can be proved to be true regardless of the researcher and the occasion.

Most research methodologies are criticised in order to determine the advantages and disadvantages associated with the research methodologies. The critique should not always be seen as a negative factor. It should instead be viewed as a tool to be used in order to determine if the research methodology is appropriate for the research. The critique of positivism is discussed next.

2.1.1.4 Critique

Critique is concerned with the examination or review of works of art or literature. A positivist believes that everything can be measured and calculated, and this makes positivism inflexible. Positivism tends to be inflexible once the data collection has started and cannot be changed. This paradigm is weak at understanding human social factors (Oughlin, 2012). Positivism relies on the repetition of results, and this is not always possible. The paradigm fails to take note of the fact that not everyone sees the world in the same manner. Positivism is dependent on experience as a source of knowledge, but a wide range of concepts, such as cause, time and space are not based on experience (Dudovskiy, 2016).

Criticism of positivism, according to Oates (2015), is the following:

- **Reductionism:** It is not always possible to break complex problems down into more straightforward problems, or when the problem is broken down, the bigger picture is sometimes forgotten.
- **Repetition:** It is not always possible to repeat the study several times with different researchers.
- **Generalising:** The researcher might not always seek generalisation. The researcher might want to study the particular and uniqueness of something.
- **World view:** Everyone does not have the same world view. People see the same thing in different ways.

Even though there is critique against the positivistic paradigm, it is still the appropriate research methodology to use in this study. The disadvantages do not overshadow the advantages of the paradigm. In the next section, the methods used in positivistic research are discussed.

2.1.1.5 Methods

Methods are research strategies that are followed to collect evidence for the building and testing of theories and systems. Positivism methods include descriptive research which is anything that is a variable and varies to a defined degree and can be measured in the process. Descriptive research includes the following: surveys, case studies, causal-comparative studies, correlation studies, development studies, and trend studies. The other part of positivism is based on experimental research, and this includes the deliberate use of certain factors under highly measured situations (Feigl, 2015).

The following three scientific techniques are used in positivism (Oates, 2015):

- **Reductionism:** It is the process of breaking complex problems into smaller, more easily studied problems.
- **Repeatability:** The researcher does not rely on the results of just one experiment. The experiment will be done several times to ensure that the same results are obtained. Other researchers will also try to repeat the experiment to determine if they will obtain the same results and to ensure that the original researcher did not influence the results that were obtained.
- **Refutation:** If other researchers cannot repeat an experiment and obtain the same result, they will refute the hypothesis. The hypothesis can also be refuted if a researcher can show that the experiment which claims that A causes B is not valid under a specific circumstance.

The methods identified above help with breaking the problem into smaller, more manageable pieces. Simplifying UAS control and improving automation can be viewed as a complicated

problem. Using the methods mentioned above, the problem is simplified by identifying the various aspects involved in this specific problem. For example, arising from the problem it can be inferred that various existing UAS control techniques should be investigated to determine solutions to simplifying UAS control. Next, the data gathering techniques of positivism are discussed.

2.1.1.6 Data gathering

Data gathering is the systematic approach of collecting information from a variety of sources to get a complete and accurate image of the area of interest (Rouse, 2015). Positivism relies on quantitative data collection. Quantitative data is collected and recorded systematically in a database for analysis. Quantitative data involves the use of numbers to assess information and the information can be evaluated, using statistical approaches which allow the researcher to find the meaningfulness of the data (Degeling, 2010). Experiments are one of the methods used to gather data when using the positivistic approach. When an experiment is conducted, the researcher makes precise and detailed observations of the outcomes and changes that occurred when a particular component is introduced or removed (Oates, 2015). In this study, quantitative data is collected through various experiments for analysis. The data analysis technique of positivism is discussed in the next section.

2.1.1.7 Data analysis

Data analysis is the process of evaluating and processing data, using analytical and logical reasoning. The data is analysed using mathematical and statistical approaches. This allows the researcher to look deeper into the data and to find the information which is needed. It is essential to analyse the data before a conclusion is generated. Positivists acknowledge that multivariate analysis can form fundamental connections among two or more variables (Trueman, 2015). This approach will be used to distinguish between different hand gestures in this study.

The positivistic paradigm focuses on tests and experiments that can be controlled and measured. The results that were obtained can be used as a reference when the experiment is repeated and to ensure that the same results are obtained. In the next section, a discussion on the research strategy is presented.

2.2 Research strategy

A research strategy is a step-by-step guide that gives direction and enables the execution of research in a systematic and scheduled manner to produce quality results (Johannesson &

Perjons, 2014). The research strategy enables the researcher to stay focused, reduce obstacles, enhance quality, and save time and resources.

2.2.1 Design science

The design science research strategy focuses on the development of new information technology products, also known as artefacts (March & Smith, 1995). This research strategy enables the researcher to offer a construct, model or instantiation as a contribution to knowledge (Oates, 2015). Oates refers to design science as design and creation, but by investigating the sources Oates used, it can be seen that the sources refer to the term as a Design science. This strategy is ideal for this study. The tool used in design science helps to identify the method that works best. The method focuses on first creating a prototype, evaluating the prototype and then repeating the process to identify the best methods to use. In the next section, the framework of design science is discussed.

2.2.1.1 Design science framework

Every type of research has certain specific procedural guidelines. Several factors apply to most methods used for academic research. The research must be motivated by a problem that is suitable for the form of research being conducted (Ellis & Levy, 2008). The research must be based on research questions that can be answered by the sort of research being conducted. The research must recognise the assumptions, boundaries, and delimitations upon which the research is based. Results from the research can only be attainable by the approaches being used. The results of the research must support the conclusions. In the next section, the major steps of design science research are discussed.

2.2.1.2 The major steps of design science research

There are several different names and number of goals in literature when it comes to design science research (Nunamaker *et al.*, 1991). Design science is a problem-solving approach. It uses an iterative process which involves the following five steps (Vaishnavi & Kuechler, 2004): awareness, suggestion, development, evaluation, and conclusion. The details of these steps are as follows:

- **Awareness:** One of the essential aspects of using design science research is to identify the problem (Nunamaker *et al.*, 1991). Not all problems that are identified are research worthy, and not all problems are appropriate to use with the design science research methodology (Ellis & Levy, 2008). Several different problems can drive design science studies. Newly emerging or evolving situations often create situations in which there is no product, tool, or model available to address this problem. Problems can be identified

by studying literature where the researcher identifies areas for further research, reading about discoveries in another field of study, finding a need for something, or from new development in technology (Oates, 2015). The output for this phase is a proposal for a new research effort (Vaishnavi & Kuechler, 2004).

- **Suggestion:** Suggestion is the identification of a tentative idea that might solve the problem (Oates, 2015). A suggestion is an inventive step where new functionality is envisioned based on a new configuration of either current or new elements (Vaishnavi & Kuechler, 2004). The suggestion step has been criticised for bringing non-repeatability into the design science research method, as human creativity is still perceived as a poorly understood cognitive process. In positivistic research, creativity is the fundamental leap from the curiosity about a specific phenomenon. This leap is used for the development of an appropriate construct that operationalises the phenomenon. It is also an appropriate research design for the positivist measurements.

- **Development:** In this phase, the tentative idea is further developed and implemented (Oates, 2015). How this will be implemented depends on the kind of artefact being proposed. An algorithm may require the construction of formal proof to show that the algorithm is correctly implemented (Vaishnavi & Kuechler, 2004). An expert system using new assumptions may require software to be developed.

- **Evaluation:** Once the artefact has been created, it is evaluated according to the criteria that have been mentioned in the proposal of the research project (Vaishnavi & Kuechler, 2004). An assessment is made of its value and how it deviates from expectations (Oates, 2015). Qualitative and quantitative deviations from expectations should be carefully noted, and tentatively explained (Vaishnavi & Kuechler, 2004).

- **Conclusion:** In this phase, the result obtained from the design process is consolidated and written down. The knowledge that was gained is also written down together with any loose ends or unexpected or anomalous results that were identified which cannot be explained yet or could be subjected to further research (Oates, 2015).

The design science phases are not followed in a rigid, stepwise manner. Design science uses a more fluid, iterative cycle (Oates, 2015). This method gives greater awareness of the nature of the problem when the researcher thinks about tentative solutions to the problem. A design science strategy thus enables the researcher to learn through making. The following section entails a discussion on the development methodology that is used in this study.

2.3 Development methodology

Software development methodologies or system development methodologies is a software engineering framework that is used to structure, plan, and control the process of developing an information system (Gechman & Gechman, 2019). In this section, a brief discussion of the agile development methodology and Scrum, as an agile development methodology are presented. There is also a discussion on why Scrum was used in this study at the end of this section.

2.3.1.1 Agile development methodology

The agile software development method has become more popular during the past few years and has been accepted among mainstream software developers since the late 1990s (Begel & Nagappan, 2007). The increasing interest in agile development is due to the belief that agile methods are powerful development alternatives and can be better at avoiding project problems including low productivity, schedule delays, high costs, and lack of motivation (Cardozo *et al.*, 2010). Agile development is based on an iterative approach that is used for the development of software. The process is iterative, and this allows for software to be built in progressively small pieces, with each piece adding to the features of the preceding iteration (Williams, 2002). If for example, iteration *A* is completed and new information is found, it could lead to requirement adaption for iteration *A* + 1. A scope is chosen for each iteration to fill the iteration length. The iteration length stays the same for the chosen scope, but the scope is reduced to fit the iteration length. The main difference between agile and previous iteration methods is the length of each iteration. Scrum is agile and used for project management.

2.3.1.2 Scrum methodology

Scrum is one of the most studied agile methods for its novelty and can improve project productivity (Cardozo *et al.*, 2010). Scrum uses an empirical process based on flexibility, adaptability, and productivity.

The characteristics of the Scrum methodology, according to Schwaber (1995) are as follows:

- The planning and closure phases consist of defined processes, where all the processes, inputs and outputs are well defined. Knowledge of how these processes will be completed is explicit. There are a few iterations in the planning phase, but the general flow is linear.
- The sprint phase is an observational process. Many processes in the sprint phase are unknown or uncontrolled. It is treated as a black box that requires external controls.

- Sprints are flexible and nonlinear. When it is possible, explicit process knowledge is used. If explicit knowledge cannot be used, tacit knowledge and trial and error are used to build process knowledge.

Scrum has the following phases:

- **Planning:** Defining a new release based on current knowledge, along with the estimated schedule for completion. If a new system is being developed, this phase consists of conceptualisation and analysis. The limited analysis is used when an existing system is being enhanced. In this study, the planning phase is initiated by identifying the problem. The literature review forms part of the planning phase, and from the deliverables of the literature review, planning begins on the development of the system.
- **Architecture:** Design of the implementation of items that can be completed in one sprint. The architecture phase also includes system architecture, high-level design, and forms the backbone of this study. This phase assists with the identification of platforms and development environments.
- **Development sprints:** The development of new functionality, by considering variables of time, requirements, quality, cost, and competition. There are many iterative development sprints or cycles that are used to develop the system. Each problem that needs to be addressed in this study forms part of a sprint. Each sprint follows on the previous sprint.
- **Closure:** Preparation for release, including final documentation, testing, and release. In the closure phase, the data that was collected will be analysed to determine if the solution is viable.

Scrum has several controls that are used in various phases. Management can use these controls to manage backlogged items. Teams use these controls to manage changes and problems.

Scrum has the following controls (Schwaber, 1995):

- **Backlog:** The functionality requirements of the system which are not adequately addressed by the current product release. The following are backlog items: Bugs, defects, customer-requested enhancements, competitive product functionality, competitive edge functionality, and technology upgrades.
- **Release/Enhancements:** Backlog items that will at a point in time serve as an appropriate release, based on the variables of requirements, time, quality, and competition.

- **Packets:** The components or objects which need to be changed in order to implement a backlog of items into a new release.
- **Changes:** The changes that need to occur to a packet in order to implement a backlog item
- **Problems:** The technical problems that could occur and that must be solved to implement a change
- **Risk:** Risks are continuously assessed, and responses are planned to ensure the success of the project.
- **Solutions:** Some solutions often result in changes to the project.
- **Issues:** These are other project issues that are not defined in terms of packets, changes and problems.

These controls are reviewed, modified, and reconciled at every sprint meeting. Scrum uses sprints which are discussed below.

Sprints - Scrum uses sprints which are a set of development activities that need to be conducted over a pre-defined time, usually within one to four weeks (Cardozo *et al.*, 2010). The content of the interval is based on product complexity, risk assessment, and the degree of oversight which is desired. The pre-determined duration of the sprint drives the intensity and speed of the sprint. Risk is continuously assessed, and adequate risk controls and responses are put in place. A sprint is used to produce a visible, usable, and deliverable product that implements one or more user interactions within the system (Rising & Janoff, 2000). The main idea behind each sprint is to provide valuable functionality to the system. Each increment builds on the previous increment.

Each sprint consists of the following (Schwaber, 1995):

- **Develop:** Defining the changes that are needed for implementing backlog requirements into packets, opening these packets, performing domain analysis, designing, implementing, testing, and documentation. The development consists of the following micro-processes: discovery, invention, and implementation.
- **Wrap:** Completing the packet, creating an executable version of the changes and how they implement backlog requirements
- **Review:** A meeting is held to present work and review progress, raising and resolving issues and problems, adding to new backlog items. Risk analysis is also conducted, and appropriate responses are defined.
- **Adjust:** Applying the information that was gathered at the review meeting to the affected packets, including a new look, feel, and properties

2.3.1.3 The use of Scrum in this study

The systems implemented in this study are completed in several phases, with each phase building on the previous phase. As mentioned, Scrum is agile and agile development is an iterative process. Complex problems are broken into smaller, more manageable problems with the help of reductionism. Reductionism is one of the methods used in positivism. By breaking the problems into smaller problems, it is easier to apply the Scrum technique. As stated, Scrum uses sprints, and it is an appropriate technique to combine with the process of reductionism. For example, the process of using hand gestures to control the UAS can be seen as a complex problem. There are various processes involved which are described in more detail in Chapter 4. The necessary process is to have communication between the smartwatch and the smartphone and then communication between the smartphone and the UAS. While these devices are communicating, data needs to be captured and processed by a neural network. It is relatively easy to see that it becomes a complex problem. However, by using reductionism, it can become more straightforward; for example, the process of communication between the smartphone and smartwatch can be seen as one problem. By using Scrum and sprints, the first sprint can be used to solve this problem and then the next sprint can be used to solve the process of having communication between the UAS and smartphone. Each sprint then builds on the previous sprint, and the process of problem solving becomes simpler. Another reason for using Scrum is that although this is an individual project, it could form part of a more significant project, and by using Scrum, it becomes simpler to share information. Other researchers could need the same techniques that were used, and because Scrum encourages short daily meetings, it can become a space to share knowledge.

2.4 Conclusion

In this chapter, the focus was on the research paradigm, research strategy, and development methodology that are followed in this study. Each of these research method characteristics is used throughout this study to aid in the process of finding a viable solution to address the problem statement. In the next chapter, a literature review is given, based on previous research on UASs and techniques that can be used to simplify control and improve the automation of a UAS.

CHAPTER 3 LITERATURE REVIEW

In this chapter, the literature review is given. Various topics are discussed in the literature review, each aiding in the process of solving the problem statement addressed in this study. A UAS is more than just an unmanned aircraft (Klimkowska *et al.*, 2016). The main components of the system consist of the unmanned aircraft, communication systems, payload (sensors, cameras, packages, or any additional equipment carried by the UAV), ground control station, recovery and launch equipment, as well as support equipment. The end goal is to automate the UAS, as defined by the problem statement in Chapter 1. Automating the UAS requires the investigation of various topics. The first topic, discussed in Section 3.1 is the unmanned aerial vehicle and its components which includes the aircraft, sensors, and actuators. In Section 3.2, the different types of control system for a UAS are discussed. In Section 3.3, the information gathered in the previous two sections is brought together, and techniques are investigated on automating UAS control. The content in these sections is used for automating the UAS, but it does not cover all the subjects that are involved in a UAS. Only the required UAS automation-related topics for this study are covered. Artificial neural networks in general, the radial basis function neural network and the Fast Fourier Transformation, are also discussed. These techniques enable the recognition of gestures and activities performed by the pilot. In Section 3.4, the support tools, such as algorithms and system development tools are investigated. The chapter is concluded in Section 3.5.

3.1 Unmanned Aerial Vehicles

UAVs are power-driven, reusable aeroplanes operated without a human pilot onboard (Chao *et al.*, 2010). The first UAV was named the Q-2 and was made by Ryan Aeronautical. It was used in the 1950s for the military observation of a region in order to locate an enemy or discover strategic features (Herwitz *et al.*, 2004). There are different UAVs available on the market, making it easier to choose one that is suitable for mission purposes (Klimkowska *et al.*, 2016). UAVs can be classified into four major categories based on the UAV's wing shape and body structure. These UAV characteristics are briefly discussed below:

- **Fixed-wing:** These UAVs are unmanned aircraft with fixed wings. They are mainly used for aerial mapping and inspection of pipelines or power lines (Tahir *et al.*, 2019). The main advantage of the fixed-wing UAV is that it has a simple structure (compared to a rotary wing) and is easy to maintain and repair (Klimkowska *et al.*, 2016). The characteristics of the fixed-wing allow for longer flight duration at higher speeds which enables the surveying of larger areas. Fixed-wing UAVs are capable of carrying a large payload for longer distances. It enables the ability to attach bigger sensors to the

aircraft. One of the disadvantages of a fixed-wing UAV is that it needs a runway or catapult for take-off and landing. Another disadvantage is that the aircraft needs to be in a constant forward motion to enable the wings to produce lift.

- **Flapping-wing:** The UAVs have small and flexible wings and are based on flying techniques used by insects and birds (Klimkowska *et al.*, 2016). The primary motivation behind flapping-wing UAVs was the need for aerial surveillance inside a building that has confined spaces (Mueller, 2001). Flapping-wing UAVs' movement has proved to be more stable than quadrotor UAVs that rely on rapid wing rotations (Shyy *et al.*, 2010).
- **Rotary-wing:** These UAVs have two or three rotor blades that revolve around a fixed mast known as a rotor (Klimkowska *et al.*, 2016). There are several different rotary-wing UAV configurations available (the number of rotors is indicated in brackets): helicopter (1), bicopter (2), tricopter (3), quadcopter (4), hexacopter (6), and octocopter (8). All these UAVs have their unique characteristic advantages and disadvantages. Rotary-wing UAVs work similarly to fixed-wing UAVs with the advantages of not requiring constant forward movement to generate airlift. The blades on a rotary-wing UAV are in constant movement which leads to the generation of the airlift. The most significant advantage of using a rotary-wing UAV is that the aircraft has the ability of vertical take-off and landing. Rotary-wing UAVs are a complex mechanical and electronic construction which leads to more complex maintenance and repair processes.
- **Blimps, balloons, and kites:** They are characterised as being lighter than air, can endure long distance at low flying speeds, and are relatively big in terms of size. These forms of aerial vehicle are worth mentioning to show that there are also unmanned aerial vehicles that can be operated without the use of a motor.

The most commonly used UAV type is a quadcopter (Tahir *et al.*, 2019). The quadcopter is a vertical take-off and landing platform that belongs to the rotary-wing UAV family (Criado & Rubio, 2015). A quadcopter has four brushless rotors (they use magnets to generate power) that generate the lift force, which allows the quadcopter to increase its maximum payload capability. The movement of a quadrotor UAV is further discussed in Section 3.1.3. Quadrotor UAVs are relatively inexpensive and easier to build than the other UAVs mentioned (Tahir *et al.*, 2019).

The quadcopter UAV is used in this study and the discussion on UAV sensors and actuators that follows will specifically focus on quadcopter UAVs. UAVs are flexible, as seen from the various applications (as discussed in Chapter 1) for which they are used. The various sensors and actuators associated with UAVs will be investigated to identify the tools that can be used to aid with automating and simplifying UAS control.

3.1.1 UAV sensors

For the UAV to maintain a stable and controllable flight, various sensors are required (Tahir *et al.*, 2019). Sensors and sensing strategies enable a UAV to measure aspects of its surroundings (Valavanis & Vachtsevanos, 2014). Sensors provide technologies that are essential when it comes to autonomous UAV flights. Most applications of UAVs (as discussed in Chapter 1) rely on the sensors to sense the UAV's surroundings and relay the measurements back to the pilot. Some pre-defined actions might be taken according to specific measurements. In the next sections, some of the sensors that can be attached to a UAV are discussed.

3.1.1.1 Global Navigation Satellite Systems sensors

At the centre of most UAV navigation and guidance systems is the GNSS (Global Navigation Satellite System) (Grewal *et al.*, 2007). The GNSS receiver receives and processes satellite signals to determine the distance between the receiver and satellites (Zhang & Hsu, 2018). The result is then used to determine the position of the UAV. The primary purpose of the GNSS is to provide the means for flight planning, tracking and UAV localisation (Tahir *et al.*, 2019). Within a GNSS is an INS (Inertial Navigation System) and GPS (Global Positioning System). These systems' interconnected nature has been recognised, and as a result, are the preferred sensor pair used in the majority of autopilot systems. Many autonomous landing systems for UAVs are based on a GPS (Merz *et al.*, 2006). The GPS and INS sensors are not the only two sensors that are used for UAV navigation. Altimeters (laser-based and barometric) are also used with a GPS and INS to enhance the estimation of the UAV's position (Grewal *et al.*, 2007). Infrared attitude sensors (discussed in Section 3.1.1.2) can also be used and are more familiar with micro-UAV applications.

Researchers have also experimented with the integration of several other sensors in combination with GPS and INS sensors, for example, a GPS with computer vision (Dusha & Mejias, 2008) and an INS with computer vision (Merz *et al.*, 2006). In the next section, there is a discussion on the different electro-optical sensors that are used on UAVs.

3.1.1.2 Electro-optical Sensors

UAVs may use electro-optical (EO) sensors (Mejias *et al.*, 2015). These sensors have become standard onboard many modern aerial vehicles. EO sensors can be used for navigation or surveillance and are one of the most common sensors found on UAVs. Some of the EO sensors found on UAVs are as follows:

- A **visible spectrum** camera usually operates at a wavelength of about 390 nm-750 nm (Mejias *et al.*, 2015). Visible spectrum cameras can be divided into two main categories:

digital still cameras and machine vision cameras. Digital still cameras are used for high-resolution images, but cannot efficiently be used for a continuous stream of images. The number of images that can be taken by the digital still camera depends on the size of the internal memory. Digital still cameras are mainly used for remote sensing and aerial photography. Machine vision cameras operate at a relatively lower resolution than digital still cameras, but can provide a continuous stream of images up to a few hundred frames per second. The resolution and choice between digital or analogue output influence the speed. Machine vision cameras are suitable for tasks that require a fast perception of the environment.

- **Infrared cameras** detect light and convert the light in the same way as conventional visible spectrum cameras, but are sensitive to light at longer wavelengths (Mejias *et al.*, 2015). The camera forms an image using infrared radiation at wavelengths from 5000 nm-14000 nm. Infrared cameras are used to measure the thermal radiation of bodies. The intensity of each pixel is used to determine the temperature of the body.
- **Hyperspectral imaging** sensors acquire data about images in multiple in-line spectral bands. Hyperspectral sensors generate a high volume of data that needs to be processed and interpreted according to specific properties and how they relate to the actual measurement made from the sensors. For example, a single cell's position in an image will have a set of intensity or brightness levels for each of the wavelengths. The hyperspectral sensor examines different materials and identifies different intensity wavelength relationships, based on prior knowledge. The image data can then be used to identify the type of material.

These sensors detect and process wavelengths in, or close to the visible part of the electromagnetic spectrum to create a visual representation of the UAV's surroundings. EO sensors rely on vision to gather information. It is therefore necessary to investigate other sensors that can be used when optical vision is not available. In the next section, a discussion on radio wave sensors is provided.

3.1.1.3 Radio wave sensors

Radio wave sensors use radio waves to gather information and can aid with the UAV's navigation in the air even when vision is weak or not available. Radio wave sensors that can be found on UAVs are as follows:

- **Airborne radio detection and ranging (Radar):** This is a radio system and is used to determine the range, altitude, direction, and speed of objects (Mejias *et al.*, 2015). The system transmits controlled radio pulses that are reflected by the object, and the distance between the system and object is determined by the time it takes for the signal

to return. Speed can be determined by tracking the change in distance with time or by using the Doppler effect (Stimson, 1998). Traditional radars in UAVs have one main drawback in that they consume a large amount of energy because of the weight and size (Mejias *et al.*, 2015). New systems, such as the Synthetic-Aperture Radar are beginning to make onboard radar technology feasible UAVs (Hanlon, 2008).

- **Light Detection and Ranging (LIDAR):** LIDAR uses a similar technique to radar (Schwarz, 2010). LIDAR uses light pulses instead of radio waves. LIDAR estimates distance by measuring the time it takes for light pulses to reflect from an object (Schwarz, 2010). LIDAR has become popular to use with mapping and infrastructure inspection (Liu *et al.*, 2009). It is an active system that makes use of ultraviolet light to map objects without the need for external light for useful mapping (DJI, 2019).

These sensors are used to aid the UAV with its navigation through the air. In addition to sensors, various techniques are used with these sensors for various requirements (discussed in Section 3.2.3). Sensors are also used in combination with actuators which are discussed in the following section.

3.1.2 Actuators

Actuators are mechanical or electromechanical devices that are used to control something (Thomasnet, 2018). Some of the actuators that can be found on UAVs are as follows:

- The landing gear is usually found at the bottom of the UAV and aids the UAV to perform successful take-off and landing (Reagan, 2014).
- The gimbal is a camera mount that can be found on most UAVs (Reagan, 2014). The gimbal allows the camera to move along multiple axes by remote control. Most gimbals used servo motors to move the camera, but recently brushless gimbals were introduced which are described by experts as simpler and requiring less maintenance. The brushless gimbal is controlled by three motors that can keep the camera level on all axes as the operator (or the UAV) moves the camera (Trofin, 2015). An inertial measurement unit is used to respond to movement to stabilise the camera by utilising the three motors.
- There are several different types of energy sources available for UAVs. Most modern UAVs use lithium polymer batteries. These batteries are lightweight and have maximised charge capacity and power (Reagan, 2014). UAV batteries need to be small and light to push boundaries and flight performance (Drone Industry Insights, 2017). A specific limit has been reached when it comes to the power density of these batteries, and adding more batteries will not necessarily increase the flight times and payload capacity. There are a few aspects that need to be considered to achieve higher payloads and longer

flight times. Examples of these aspects include the mass-specific energy (how much power (Wh) per unit mass (kg) is needed) and the volumetric specific energy (how much power (Wh) per unit volume (m^3) is required).

- The intended mission also influences the decision for choosing the energy source. Besides, there are petrol-powered UAV solutions. The significant advantage with a petrol-powered UAV is that the UAV loses weight over time, making the UAV lighter and increasing the range. Solar power is also an option and is mainly used to increase the range of the UAV. There are various other energy sources and they all have their advantages and disadvantages (Drone Industry Insights, 2017).

These actuators are some of the most common actuators that are found on UAVs. The rotors also form part of the actuators (discussed in Section 3.1). It is necessary to understand the movement of a UAV to determine how the UAV control can be automated. In the next section, an overview of how a UAV can fly and navigate through the air is presented.

3.1.3 UAV vertical flight overview

An aircraft rotates around its centre of gravity and the principal axes of pitch, roll, and yaw (Nigel, 2017; Rodkin, 2014). Figure 3-1 is a visual representation of the six degrees of freedom. Six degrees of freedom is a specific parameter count which represents the number of degrees of freedom an object has in a three-dimensional space (Techopedia, 2019).

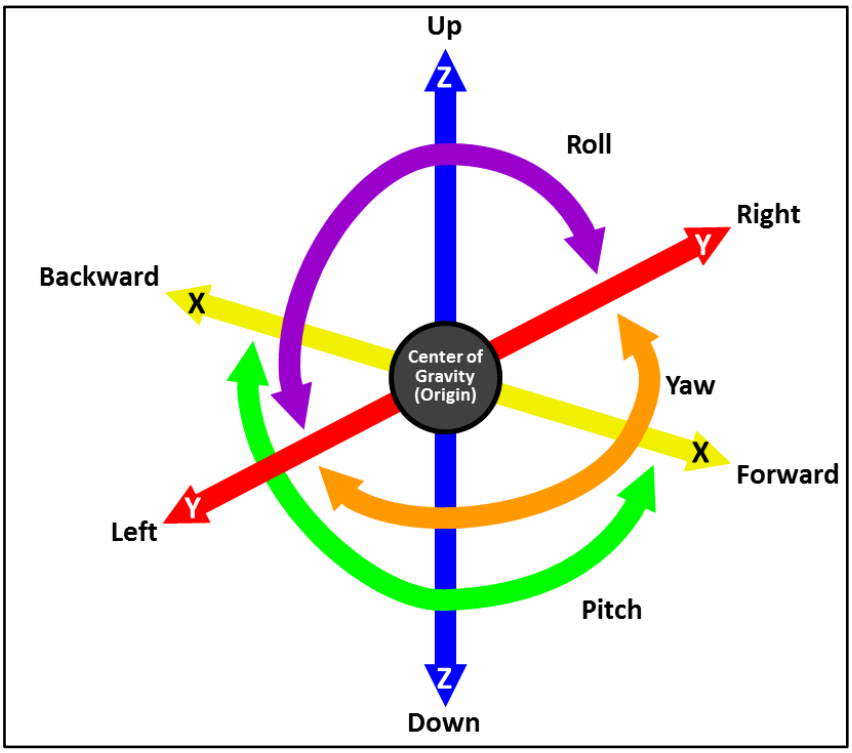


Figure 3-1: The six degrees of freedom: forward/back, up/down, left/right, yaw, pitch, roll (Nigel, 2017)

It means that there are six different ways the body of the object can move. The six degrees of freedom consists of the following movement parameters:

- **Translation**: Moving along the different axes *x, y* and *z*.
    - Surging – Moving forwards and backwards along the *x*-axis (yellow arrow).
    - Swaying – Moving left and right alongside the *y*-axis (red arrow).
    - Heaving – Moving up and down the *z*-axis (blue arrow).
- **Rotation**: Turning to face a different axis.
    - Pitch (green arrow) – Moving between the *x* and *z axes*. The aircraft moves up and down from front to back.
    - Yaw (orange arrow) – Moving between the *x* and *y axes*. The aircraft moves in a clockwise/anticlockwise rotation.
    - Roll (purple arrow) – Moving between the *y* and *z axes*. The aircraft moves left or right.

In addition to pitch, roll, and yaw, a throttle is also used to control the UAV. The throttle controls the altitude (vertical distance between the aircraft and sea level) of the vehicle, getting airborne, and speed (Nigel, 2017). To achieve these six degrees of freedom on a quadrotor UAV, the speed of the rotors needs to be adjusted accordingly. In the next section is a discussion on how a UAV adjusts the speed of the rotors to achieve movement within the six degrees of freedom.

3.1.3.1 UAV flight control overview

A quadrotor UAV has two pairs of counter-rotating rotors and propellers that are positioned at the apexes of a square frame (Lee *et al.*, 2010). The two pairs of propellers (1, 3) and (2, 4) can be seen in Figure 3-2. Unlike traditional helicopters that have different pitch angles, a UAV has fixed pitch angle rotors (Altu *et al.*, 2002). The speed of the rotors is adjusted to change the lift force and create motion (Bouabdallah *et al.*, 2004). By increasing or decreasing the speed of all four propellers at the same time, it generates vertical motion. The quadrotor UAV is capable of vertical take-off and landing and does not require complex mechanical linkages like swash plates or teeter hinges that are commonly found on helicopters (Lee *et al.*, 2010). To produce a roll rotation with lateral motion, the speed of propellers 2 and 4 needs to be changed (Bouabdallah *et al.*, 2004). The speed of propellers 1 and 3 is adjusted to get a pitch rotation. Yaw rotation is achieved by the difference in counter-torque between each pair of propellers. The quadrotor enables the aircraft to have an increased payload capacity, and with more lift, heavier weights can be carried (Altu *et al.*, 2002). Quadrotors are highly manoeuvrable, enabling flight in constrained places.

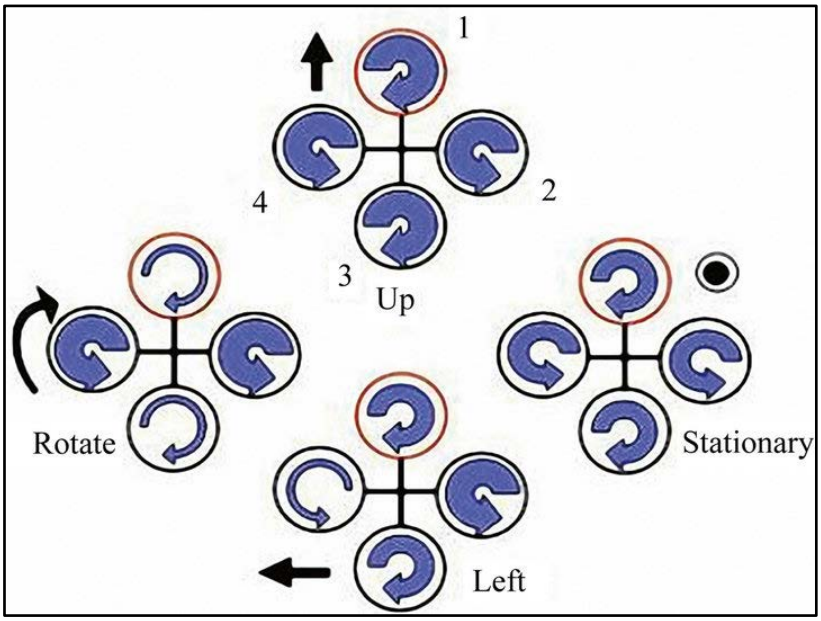


Figure 3-2: Description of quadrotor motion where the width of the arrow indicates the rotational speed (Bouabdallah *et al.*, 2004)

A UAV is commonly controlled with a remote control (Anand & Mathiyazaghan, 2016). Figure 3-3 is a visual representation of a traditional UAV remote control.

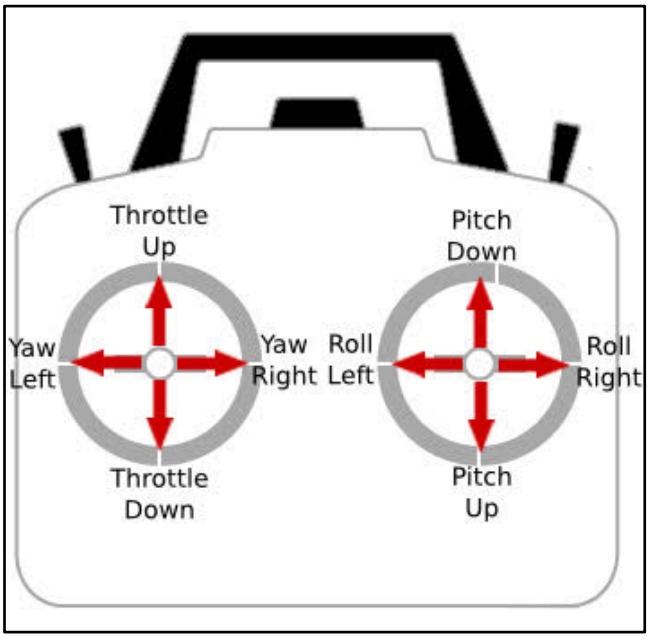


Figure 3-3: Visual representation of a traditional UAV remote control (Cho, 2016)

The UAV is controlled according to the input from the remote control by giving the appropriate throttle, pitch, yaw and roll values manually (Anand & Mathiyazaghan, 2016). The input on a remote control is given as follows (Drone Industry Insights, 2017; Hobbytron, 2019):

- **Throttle:** Push the left stick forward to increase and pull the left stick back to decrease.
- **Pitch:** Push the right stick forwards or backwards.
- **Yaw:** Push the left stick to the left or the right.

- **Roll:** Push the right stick to the left or right.

In this section, a discussion on some of the different available types of unmanned aircraft was given. The discussions that followed focused on the quadrotor UAV and the sensors and actuators associated with these UAVs. A more in-depth discussion was given about the rotor actuator and how it is used to enable the UAV to navigate through an area. These discussions focused on the UAV itself, and in the next section, a discussion on some of the different control systems that can be used to control a UAV is given.

3.2 Control systems

A control system is an interconnection of components forming a system configuration that is used to get the desired response from the system (Dorf & Bishop, 2011). There are two types of control systems, an open-loop control system and a closed-loop control system. An open-loop control system uses the controller and actuators to obtain the desired response, and the system does not provide feedback. A close-looped control system utilises an additional measure of actual output and compares it to the desired output. It is beneficial to investigate the different types of UAV control system to identify methods that can be used to simplify and improve automated UAS control. The next section is a discussion on the traditional flight control systems that are commonly found on a UAS.

3.2.1 UAS traditional flight control system

A UAV requires a flight control system to ensure stable behaviour and desired performance (How *et al.*, 2014). For example, when a UAV follows the desired trajectory, it should be with sufficient accuracy in the presence of external disturbances (e.g. trees and a building). The flight control system is architecturally mission/safety-critical, as an incorrect design can result in poor performance or even vehicle loss.

There are various methods to control a UAV manually and autonomously (Stevenson *et al.*, 2015). There are different levels of automation (discussed in Section 3.3.4). On the lowest level of automation, the aircraft relies entirely on the remote control to be piloted (Stansbury *et al.*, 2009). In the absence of a pilot, the aircraft would lose control and crash. On the highest level of automation, the aircraft is autonomously controlled by an autopilot from take-off to landing. The pilot in command only intervenes in the case of an emergency and overrides the autopilot. Most modern UASs are equipped with manual remote control and autopilot technologies. UAV controls can be divided into two sections, autonomous and manual control (Stevenson *et al.*, 2015). The most common manual and autonomous UAV control include the following:

Manual control comprises

- Radio control by an external pilot that in some cases uses a third-person remote view of the UAV;
- A flight console which is similar to a cockpit and uses a fixed forward camera view to allow the external pilot to fly the UAV as in a simulator; and
- First-person view flying that uses virtual reality and head-tracking techniques.

Autonomous control comprises

- Autopilot control, typically using GPS waypoints to define the flight plan;
- Airspeed and pressure sensors which are used for inner-loop airframe (attitude of the UAV) control; and
- Automatic take-off and landing capabilities that are offered by some autopilots.

In the next section, some of the manual control systems used on a UAS are discussed.

3.2.2 Manual control systems

The instruments that are provided for manual control of flight paths are similar to those that can be found inside the cockpit of an aircraft (Tadema *et al.*, 2007). The out-of-the-window view is represented by a camera that is usually facing forward. The richness of visual, motion and audio cues that are available on standard manned aircraft are not available to the UAS operator. Information that is available to the operator of the UAS suffers from slow update rates and low resolutions that are caused by sensor and data link bandwidth limitations. Most UAS accidents occur during the take-off or landing process, especially by a UAS that relies on manual piloting to accomplish these tasks (Stevenson *et al.*, 2015). The UAS can be piloted in third-person or first-person view.

When an external pilot controls the UAS in third-person with a radio controller, the pilot must control all three positioning angles (pitch, roll, and yaw) of the aircraft (Stevenson *et al.*, 2015). The pilot must also control the throttle to maintain the correct altitude and airspeed. Various external problems can occur when controlling the UAS, for example, spatial disorientation. During spatial disorientation, the UAS can be in an abnormal position making it difficult to control. The controls can be inverted when the UAS is in an upside-down state or the incorrect orientation. The disorientation can cause the UAS to crash, primarily due to improper control inputs. There is also the problem of the UAS flying too far away from the pilot. Finally, adverse light conditions can make it difficult to see the UAS and almost impossible to determine the orientation of the aircraft.

A UAS can also be piloted using a first-person view with the help of first-person view goggles that streams the video directly to the pilot. Pitch, roll, and yaw are now directly linked to the forward-facing view of the pilot. First-person view gives the pilot a better view of the current situation of the UAS. Next, the control of a UAS with smartphones and tablets is considered.

3.2.2.1 Smartphone- or tablet-controlled UAVs

Traditionally, UAVs are controlled with dedicated remote controllers. Modern technology has enabled the control of UAVs with smartphones or tablets. UAVs can now be controlled by using an application loaded on the smartphone and a Wi-Fi or Bluetooth connection.

There are various applications available for Android, iOS, and Windows devices that can be found on their respective application stores. FreeFlight Pro is a mobile application that is used to control Parrot drones (Parrot, 2018). Beginner and advanced pilots can use FreeFlight Pro. The application's interface can be customised to suit the skill level of the pilot. FreeFlight Pro provides advanced photo and video settings enabling photo capture and real-time video streaming. Pre-programmed autonomous flight plans can also be used with the FreeFlight Pro application. Various other UAV control applications are available for mobile devices. UAV manufacturers provide the user with the necessary documentation and software development tools to build custom applications. Smartwatches can also be used in conjunction with mobile handheld devices to control a UAV, as described in the following section.

3.2.2.2 Smartwatch-controlled UAVs

Wearable devices are widely used, and numerous companies have released various smart wearable devices, for example, smart glasses, bracelets, necklaces, and watches (Kim *et al., 2014*). Among these devices, the smartwatch is the most commonly used wearable device trend. A smartwatch not only serves as a watch itself, but the watch also provides additional features by using various sensors, such as accelerometers and gyroscope sensors. There are several sensor-related apps, such as a pedometer, heart rate monitor, and exercise assistance apps. Smartwatch technologies provide opportunities to design new forms of user interactions with mobile phones. The significant advantage a smartwatch has is its ability to provide location information (Rawassizadeh *et al.*, 2014). Like augmented reality glasses, a smartwatch does not always require both hands to operate the device. The user does not need to hold the device in his/her hands. Smartwatches enable the user to have access to multiple applications directly from his/her wrist without the need to touch his/her smartphone (Komninos and Dunlop, 2014). A smartwatch is employed in this study to control the UAV. Next, a discussion on voice-controlled UAS control methods is given.

3.2.2.3 Voice-controlled UAVs

A UAV can also be controlled by using voice commands (Anand & Mathiyazaghan, 2016). Draper *et al.* (2003) implemented a speech input system to control a UAV. The input of speech was achieved with Nuance. Nuance is a speaker-independent continuous speech recognition system that supports dynamically extensible grammars. The vocabulary of the experiment contained 160 words. A “push-to-talk” button activated the speech recognition system. From the results, it could be seen that the speech input was significantly better than the manual input. In this study, a system similar to the “push-to-talk” button to capture data is used. This system will be explained in more detail in Chapter 4.

Chandarana *et al.* (2017) also implemented a speech recognition system. The system used a commercial off-the-shelf headset microphone with speech-to-text software. The user specifies the desired control by speaking into the microphone. The speech is broken into phonemes or small and distinct units of sound that usually correspond to consonants and vowels which are then compared to the application’s dictionary of phonemes and are then mapped to the different control formations. Gesture control is discussed next. It will be necessary to undertake an overview of gesture control, as it is one of the techniques implemented in this study to improve UAS control.

3.2.2.4 Gesture-controlled UAVs

Chandarana *et al.* (2017) used a commercial off-the-shelf Leap Motion Controller sensor to control a UAV. The Leap Motion device is placed in front of the user with the assumption that the user will be performing gestures with his/her right hand. Gesture movements performed by the user are used to determine the movement of the UAV. The Leap Motion sensor uses a natural interface that allows the user to present movements by merely imitating the shape. Input gathered from the Leap Motion is characterised by using a linear support vector machine model that was trained to recognise gestures. For each gesture, the movement of the user’s palm is tracked for three seconds. Eigenvalues and direction of the movement throughout the gesture are then used to extract raw data which are classified by the trained model. In another study, a smartwatch is worn on the user’s wrist (Xu *et al.*, 2015). Data gathered from the smartwatch can then be used to identify the user’s arm, hand and possibly finger movements. The data is acquired by accessing accelerometer and gyroscope sensor data.

Various other manual control systems can also be used to control a UAS. Smartphones and smartwatches are more readily available, making it easier to use these devices to improve UAS control. In the following section, a discussion on autonomous UAS control is given.

3.2.3 Autonomous UAV control systems

A UAS can be considered autonomous when the UAS can make decisions and react to events without the direct intervention of a human pilot (Sebbane, 2015). There has been a significant amount of research done related to autonomous flight control of a UAS (Bry *et al.*, 2012). It can be a challenge to operate a UAS autonomously (Rafi *et al.*, 2006). Long-distance navigation and waypoint following are considered to be an easier automated task for a UAS. Close-range navigation and following a moving target (discussed in Section 3.2.3.6) are more challenging. Close-range navigation requires constant real-time decision making and the optimisation of many parameters that takes care of the physical constraints of the aircraft. It becomes even more difficult for the UAS to follow a moving target if the future position of the target is not known.

The most crucial research goal of autonomous navigation and manoeuvring is to reduce the time and requirement for human interaction to operate the autonomous system (Rafi *et al.*, 2006). Some advantages of automating control are that it increases the exploration capabilities at a lower risk and reduces cost in terms of time and money. The goal of most researchers is to reduce the human and UAS interaction ratio so that fewer human interactions are needed to fly the UAS and human decisions can be changed into advanced or policy level manoeuvres. Several fundamental aspects are common to all autonomous vehicles. These aspects include the ability to sense and perceive the environment, analysing (the sensed information), communicating, planning, decision making, and acting based on these findings, using control algorithms and actuators. A few of the actuators were mentioned in Section 3.1.2. Some of the control algorithms that are used with actuators for autonomous UAS control are discussed next.

3.2.3.1 Range sensor

One control approach is to use range sensors, such as laser range finders, infrared sensors, or red, blue, and green depth sensors (Padhy *et al.*, 2018). Bry *et al.* (2012) presented a state estimation method that was based on an inertial measurement unit and a planar laser range finder. The algorithm that was used is capable of maintaining an accurate state estimate of the air vehicle during aggressive flights in an unstructured three-dimensional environment without the use of an external positioning system. The algorithm is based on an extension of the Gaussian Particle Filter (GPF). The GPF partitions the state according to the measurements' independence relationship. A pseudo-linear update is then calculated to allow for the use of 20 times fewer particles than a native implementation which can achieve similar accuracy in the state estimate. Roberts *et al.* (2007) used a single ultrasonic sensor and four infrared sensors for collision avoidance, stability control, and anti-drift control for autonomous flights. These sensors are not standard on most UASs, as the onboard device is often too heavy for the UAS.

Next, a discussion on the simultaneous location and mapping algorithm implementation is provided.

3.2.3.2 Simultaneous localisation and mapping

A 3-D map of an unknown indoor environment can be inferred by using range sensors or visual sensors, while simultaneously estimating its position on the map (Checchin *et al.*, 2010; Engel *et al.*, 2014; Mei *et al.*, 2011). Bachrach (2009) used a laser rangefinder sensor for high-level simultaneous location and mapping (SLAM) implementations and exploring unknown indoor environments. Çelik *et al.* (2009) presented a novel approach to an indoor navigation and ranging strategy by using a monocular camera. The algorithm which was proposed is integrated with a SLAM with the focus on indoor air vehicle applications. The proposed algorithm was validated by using a self-contained micro-aerial vehicle with onboard image processing and SLAM capabilities. The range measurement strategy was inspired by vital adaptive mechanisms for depth perception and pattern recognition found in humans and intelligent animals. The navigation environment was unknown and GPS-denied. The results of this experiment showed that the system is only limited by the capabilities of the camera and the availability of good corners. In the next section, a discussion on stereo vision is provided.

3.2.3.3 Stereo vision

The relative position and accurate depth estimation of a UAV are possible with the use of stereo cameras (Bachrach, 2009; Fraundorfer *et al.*, 2012)**.** Fraundorfer *et al.* (2012) implemented an autonomous vision-based quadrotor micro-aerial vehicle system that maps and explores unknown environments. The algorithms that were necessary for autonomous mapping and exploration ran onboard the micro-aerial vehicle. The quadrotor used a front-facing camera as the primary exteroceptive sensor with the Vector Field Histogram+ algorithm for local navigation. Bills *et al.* (2011) used a monocular camera to find vanishing points. These points were used to fly the aircraft in a corridor environment. For staircases, the centre of the staircase was found. Additionally, a front-facing short-range sensor was used to avoid collisions in corners and unknown environments. Next, a discussion on the imitation learning strategy is provided.

3.2.3.4 Imitation learning strategy

Ross *et al.* (2013) applied a novel imitation learning strategy by implementing the DAgger algorithm. The DAgger algorithm trains a policy that mimics the expert's behaviour through multiple iterations of training. In this approach, the algorithm learned the controller policy that mimics a human pilot's choice of actions from demonstrations of desired behaviours. Padhy *et al.* (2018) proposed an efficient system in which a quadcopter autonomously navigates indoors

and finds a specific target by using a single camera. A deep learning convolutional neural network is used to learn a control strategy that imitates an expert pilot's choice of action. Neural networks are discussed in more detail later on in this chapter. Object detection can also be used in the process of automating UAS control. Object-detection applications are discussed in the next section.

3.2.3.5 Object detection (visually)

Object detection with a UAS has become an essential aspect of the development of an autonomous UAS for rescue and surveillance missions (Gaszczak *et al.*, 2011). Object detection is an essential task for object tracking, since it is continually applied in every frame (Porikli & Yilmaz, 2012). One method that is used for detecting moving objects is by using temporal information that is extracted from a sequence of images. Temporal information can be utilised, for example, by computing the inter-frame difference, studying a static background scene model and comparing it to the current scene, or by finding high motion areas. Another method for object detection is to slide a window across the image and classify each of these local windows as containing the target or background. Alternatively, a local interest point is extracted from the image and each of these regions around the points can be classified, rather than observing all the possible sub-windows. Usually, when an object has been detected, it is required to track the object in order for the UAS to autonomously follow the object. Object-tracking applications are discussed next.

3.2.3.6 Object tracking (visually)

Object tracking is a fundamental task in several areas of research (De Almeida Maia *et al.*, 2016). The purpose of object tracking is to provide the object's position over time to enable the system to analyse the object's behaviour.

UASs have various sensors and actuators that can be used for autonomous object tracking and following (Bartak & Vykovsky, 2016). The UAS takes a snapshot with the camera after which the user marks an area as the object of interest, and the UAS will then autonomously follow the selected object when the object moves. The critical part of object tracking is the detection of the tracked object in a video stream that is captured by the camera of the UAS. The UAS then navigates, based on the target's exact location.

There are essentially two approaches that are used to follow a target within a video stream. The one approach uses the known location of a target from a single video frame by estimating the motions between successive video frames and computing the next position of the target. This method is generally used for object tracking and does not require much information about the target since it uses general techniques to process pictures. This method is prone to error, and

the precision of the method deteriorates with time, as the UAS will drift away from the real trajectory of the object that is being tracked. The method also fails to rediscover the object again when it loses the object that was being tracked.

Another approach for object tracking is to use techniques such as template matching, feature detection, and matching on a static picture. The advantage of this method is that whenever the object of interest that needs to be tracked appears in the picture, it can be detected. The disadvantage of this approach is that to learn what the target looks like, it requires extensive training. It is sometimes required to do multiple-object tracking. Some of the techniques that can be used are discussed next.

Multi-object tracking: Tracking multiple objects can be decomposed into two separate steps that address independent problems (Berclaz *et al.*, 2011). The first step is the use of time-independent detection, where a prediction scheme is used to infer the number and location of the targets from the available signal at every time step independently. It typically includes either a generative model of the signal given the target presence, or a selective machine learning-based process. The second step uses modelling detection errors and target motions to link object detections into possible trajectories. It is also essential for the UAS to know what to do when the object is obstructed. Techniques that can be used to track obstructed objects are considered next.

Tracking obstructed objects: One of the main problems that needs to be solved when tracking moving objects is to separate the change in the image caused by the movement of the UAS from the change in the image movement caused by dynamic objects (Rodríguez-Canosa *et al.*, 2012). One approach to solve this problem is to use a method called tracking-learning-detection and decomposing the task of long-term object tracking into three subtasks: tracking, learning, and detection (Bartak & Vykovsky, 2016). This approach acknowledges the fact that object tracking and detection become prone to error when these methods operate on their own. The tracking method provides the detector with learning data in real-time, and it enables the detector to reinitialise the tracker in the event the object gets lost. The learner compensates for the errors that will be made by the detector and updates the model to avoid the errors from being made in the future. As mentioned earlier, the user frames the object that must be followed, for example, with a rectangle in the first image so that the UAS has the initial position of the object. The method identifies the rectangle frame of the moving object in the next image while assuming that the scaling of the rectangle may change as the UAS and object move. The tracking-learning-detection algorithm uses the Lucas-Kanade tracker that is enhanced by forward-backwards consistency checking. The Lucas-Kanade tracker improves the quality of tracking over a sequence of frames, and it self-evaluates the quality of tracking (Lucas, 1981).

Another approach is to use the Clustering of Static-Adaptive Correspondences for Deformable Object Tracking (CMT) algorithm (Chakrabarty *et al.*, 2016). The CMT tracker represents the tracked object as a set of features which correspond to key points relating the current position of a feature to its position on the original image. The CMT algorithm distinguishes between the current image, the original image, and succeeding images. It then uses both kinds of correspondence to update the object model. The CMT algorithm also introduces a tolerance parameter which allows communication between frames to be robust to change due to deformations. The CMT algorithm uses a heuristic estimate to generate values that are related to the scale and rotation of the bounding box. These values are used with the estimated values of the centres of the object being tracked to determine the outputs of the CMT algorithm. Object tracking is used for object detection and tracking. It can also be used when an object cannot be visually detected or tracked. In the next section is a discussion of techniques that can be used to track an object without the use of visual actuators.

3.2.3.7 Object-location tracking (without vision)

In the past, the location tracking was implemented by carrying a sophisticated, complex, and extensive internal navigation system (INS) onboard the aircraft. It also required frequent updates from the control system by using a communication link. Radio tracking or recognition of geographical features was used to track the aircraft. Currently, global positioning systems are used to acquire location information from a series of earth-satellites. Newer GPSs are incredibly light in weight, compact, cheaper, and gives continuous location updates which result in only needing a straightforward form of INS.

The location of a UAV needs to be available on demand at any moment in time (Austin, 2010). Location tracking may form part of all of a pre-programmed mission or be part of the return to home function. When fully autonomous flights are carried out without any communication between the aircraft and the control system, it can cause accidents. A GPS onboard a UAS can give approximate coordinates during various flight tasks (Tahar *et al.,* 2016). Knowing the position and altitude can help the pilot to handle most circumstances during an emergency, such as the loss of signal or bad weather.

When the communication between the aircraft and the control system needs to be continuous, or when there is a risk of the GPS being blocked, there are other navigation methods available (Austin, 2010). These methods include:

- **Radar tracking:** The UAV is fitted with a transponder which communicates with a radar scanner from the control station. The position of the UAV can then be identified on the radar display of the control system.

- **Radio-tracking:** A radio signal is used to carry data from the UAV to the control system. The position of the UAV is determined by the time it takes for the coded signal to travel from the UAV to the control system or *vice versa*.

- **Direct reckoning:** The position of the UAV can be determined by using computer integration velocity vectors and the time elapsed. For example, in the case that the mission is over land, and the UAV carries a camera surveying the ground, the position of the UAV can be confirmed by relating geographical features with the known position on a map.

Many other studies have been conducted to overcome bad satellite reception and the loss of signal during a flight (Tahar *et al.*, 2016). Another method that is used for dealing with this problem is to use a failsafe mode. When the UAS has a loss of signal, GPS, or low battery during the flight, the UAS will return to home. It can prevent accidents from happening, and the UAS can land safely.

There are many different types of GPS available on the market. Every different type of GPS has its strengths and capabilities to acquire signals from the satellites. It is essential to know the constellation of the satellites during the flight mission to obtain accurate coordinates. Four considerations need to be taken into account when choosing a GPS for a UAV. These considerations are compliance, accuracy, interference, and errors. Recently, new regulation compliances are considered with UAV performance and the areas in which the UAV operates. UAV operators need to ensure the UAV avoids sensitive areas, such as airports, military bases and civilian areas. The accuracy of the GPS depends on the type of signal received from the satellite platform. Cheap GPSs can only receive weak and selected signals. More expensive GPSs can support and receive many different types of signals, increasing real-time position tracking of a UAS. The required accuracy of a GPS depends on the type of application. For example, the GPS used for hobbies is not always as precise as the GPS used for military applications. Military applications may require a high accuracy rate to complete missions successfully. Signal interference during the flight might be caused by transmission lines, power systems and telecommunication towers. The operator needs to avoid potential areas that might cause signal interference. Various factors, such as wind, solar weather, electromagnetic waves and the surrounding environment can cause GPS errors. All these considerations can influence the accuracy of the UAV's position. Visual and non-visual tracking each has its disadvantages. It is beneficial to investigate methods that can be used to combine these two methods and in the process, reduce the number of disadvantages of these two methods. Next, a discussion on methods that can be used for combining visual and non-visual location tracking is provided.

3.2.3.8 Combining visual and non-visual location tracking

When a UAS navigates buildings and other obstacles in an urban environment, it can weaken the GPS signal and can even cause a complete loss of the signal. When the GPS signal is lost, the dead reckoning (the process of calculating the current position of an object by using a previously determined position) capability of the onboard INS does not allow for accurate navigation. This causes the position of the UAV to be determined visually. A vision system is self-contained, not jammable, and provides a position measurement one order of scale more accurate than a standard GPS. Limited image frame rates and sensor resolution make it difficult to estimate velocity from vision. The visually-based position needs to be combined with measurements from internal navigation sensors of the UAV. A vision-aided INS can be implemented by using a Bayesian framework and can be divided into two groups: loosely coupled and tightly coupled (Vu *et al.*, 2012). The differences are as follows:

Loosely coupled image feature-aided INS calculates the absolute position of the vehicle by using an imaging sensor and then using the computed absolute position to correct the INS. Conte and Doherty (2009) used a vision-aided INS method, based on visual odometry and georeferenced image matching. It is used when the GPS is not working. The method that was implemented used vision-aided components. A point mass filter was used to fuse the visual odometry measurements with the georeferenced image matching and an altimeter sensor to provide an absolute position. They obtained position errors that are consistently less than 2 m. Durrie *et al.* (2009) proposed a vision-based navigation system for UAVs that operate on airfields where GPSs are prohibited. The system uses a particle filter to determine the aircraft's absolute position by extracting the edges of the image and then comparing it to the predicted edges from a lane-marking database. The INS reading is then improved by using the absolute position computed from the particle filter. The vision-based navigation system that is integrated with the INS runs in real-time and has a position error consistently below 15 m.

Tightly coupled image feature-aided INS uses features of an image that are compared to mapped landmarks and are extracted in the image plane. The residual between the perceived and the forecasted feature locations together with the feature measurement function are applied to a Bayesian framework to correct the INS. Miller and Campbell (2008) used a particle filter approach to augment visual features, the GPS, and inertial navigation to estimate the position and altitude of the vehicle on a two-dimensional plane. The approach gave a positional accuracy of less than 3.5 m.

Object tracking and location tracking are an essential aspect of the process of automating UAS control. In this section, it is evident that various methods can be used to control a UAS manually or autonomously. To make a UAS fully autonomous, many technological and algorithmic

developments are still needed (Becerra, 2019). One needs to understand what a fully autonomous UAS is. These findings can then be used to identify methods and techniques that can be used to automate and simplify UAS control. In the next section is a discussion on what UAS automation is and techniques on how to automate UAS control.

3.3 Automating UAS control

The level of automation of a UAS needs to be improved (Pippin, 2015). When a UAS has a high level of automation, fewer operations are required to control the UAS. This makes the UAS cheaper and easier to deploy. Another advantage is that it enables the operator to focus more on the high-level tasks rather than lower-level mundane operations. With the automation of planning capabilities, the UAS can respond faster to operation needs. Sensor data can be processed onboard the UAS and the high-level findings can be returned to the operator. Automated UAS are less susceptible to operator errors and communication link failures. What automation is and how it is applied to a UAS are discussed next.

3.3.1 Automation

Automation is defined as a machine agent which is capable of carrying out functions that are usually performed by a human (Vincenzi *et al.*, 2015). The main goal of automation is to replace human control, planning and problem solving with autonomous devices and computers (Bainbridge, 1983). An overview of how humans process information and how it can be applied to the computerised system to improve automation must be carried out. In the next section is a discussion on human information processing and how it is applied to system automation.

3.3.1.1 Human information processing applied to system automation

According to Parasuraman *et al.* (2000), there are four stages of human information processing. The first stage refers to the acquisition and registration of multiple sources of information. This stage also includes the positioning and orientation of sensory receptors, sensory processing, initial pre-processing of data which are before the full perception, and selective attention. The second stage involves the conscious perception and manipulation of processed and retrieved information in working memory (Baddeley, 1974). This stage also includes intellectual operations such as rehearsals, integrations, and assumptions, but these operations occur before the point of decision. At the third stage, decisions are made based on intellectual processing. The final stage involves the use of an action or response that is aligned with the chosen decision (Parasuraman *et al.*, 2000). This four-stage model is a simplified version of the many components that are used for human information processing, as discovered by information processing and cognitive psychologists (Baddeley, 1974). To perform most tasks, it involves inter-dependent stages that overlap temporally within the processing of operations

(Wickens & Hollands, 2000). The four-stage model of human information processing can be applied to system functions that can be automated (Parasuraman *et al.*, 2000). Parasuraman *et al.* (2000) propose that automation can be applied to four classes of function:

1. Information acquisition
2. Information analysis
3. Decision and action selection
4. Action implementation

Each of these functions can be automated to different degrees or levels. The various levels of automation which are discussed in this section can be applied to the four classes of functions. These four classes of functions will be discussed next.

3.3.1.2 Acquisition automation

The automation of information acquisition applies to the sensing and registration of input data. This operation is equivalent to the first human information-processing stage, which supports human sensory processes (Parasuraman *et al.*, 2000). On the lowest level of such automation, it may consist of strategies which are used to mechanically move sensors to scan and observe. An example of this is a commercial air-traffic controller that uses radar to acquire information on an aircraft by scanning the sky using a fixed pattern. Moderate levels of automation at this stage could involve the organisation of incoming information according to some type of criteria.

3.3.1.3 Analysis automation

The automation of information analysis involves cognitive functions, including working memory and inferential processes. At a low level of automation, algorithms can be applied to incoming data to allow extrapolation over time or predictions. For example, predictor displays have been developed which are fitted in the cockpit of an aircraft that shows the projected course of another aircraft in the airspace nearby (Hart and Wempe, 1979). A more advanced level of automation at this stage involves integration where numerous input variables are combined into a single value. An example of this is the use of a display that has an emergent perceptual feature, such as a polygon against a background of lines (Bennett and Flach, 1992).

3.3.1.4 Decision automation

The decision and action selection stage involves the selection of decision alternatives. Automation at this stage involves altering the different levels of augmentation or replacing human selection with the decisions provided by the computer system (Parasuraman *et al.*, 2000). An example of this can be seen within an expert system which is designed with conditional logic to specify a specific decision when a particular condition is met. This stage can be compared with the decision-making stage of human performance in that the system departs

from inferences because it makes explicit or implicit assumptions about the cost and value of the different possible outcomes of the decision process and nature of these outcomes.

3.3.1.5 Action automation

The final stage refers to the execution of the choice of action. Automation at this stage involves various levels of machine chosen actions that typically replaces the hand or voice of the human. The different levels of action automation may be characterised by the relative number of manual versus automatic activities used to execute a response. An example of action automation is when the control of an aircraft is automatically transferred from one airspace sector to another by pressing a single key. Action automation includes programming agents that track the user's interaction with a computer and then execute specific sub-tasks automatically in a contextually-appropriate manner (Lewis, 1998). Automation of a UAS can be divided into the following two sections: system architecture and software architecture. Next, a discussion on automation system architecture is given.

3.3.2 UAS automation system architecture

The design of a UAS may require many different types of automated system and can be different from those used for manned aircraft. The recommended architecture that is used to support autonomous UAS operations uses the front-seat-backseat driver paradigm, which is also used by many autonomous robotic systems (Benjamin *et al.*, 2009). The front-seat driver can be seen as the flight control system and includes the autopilot, electronic system, and flight controller (Pippin, 2015). The backseat driver is the mission control computer and is more cautious and concerned with long-range planning. The main idea behind this paradigm is to distinguish between low-level and higher-level autonomous operations (the levels of automation are discussed later in this section). A high-level overview of this paradigm is presented in Figure 3-4. Within this architecture, the flight control system can be operated in a stand-alone mode by a remote-control device, a predefined waypoint route, or the mission processor can drive it. In general, the flight control system can be operated as a black box, and it does not matter where the commands originate. The mission processor also operates separately from the flight controller and has little knowledge of its internal details. The flight control system includes the autopilot, GPS, operating system, and a dedicated link to a ground station computer which is used to send waypoints and control commands to the autopilot. The flight control system also produces the control signals for the aircraft's control surfaces and throttle. The system should be able to operate autonomously and be isolated from the mission control computer by having dedicated data links and power sources. It provides the system with some fault tolerance, and it enables the system to be easily extensible.

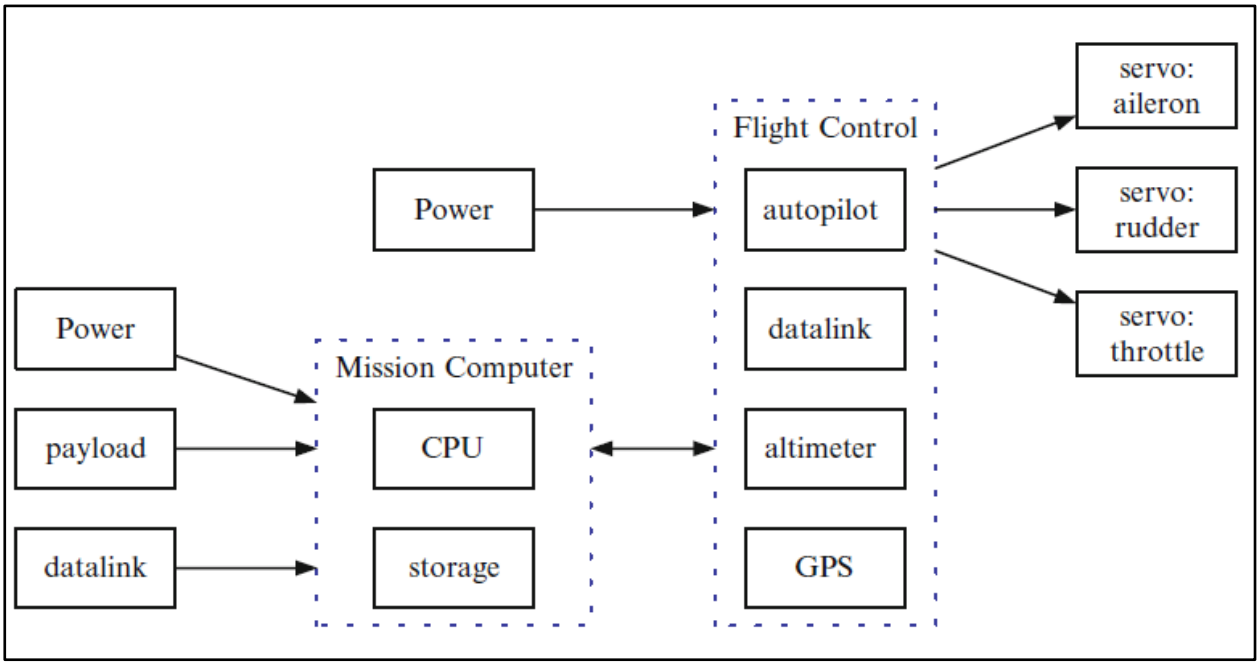


Figure 3-4: UAV system architecture (Pippin, 2015)

These components need to be easily extendable to accommodate future processing needs, autonomous behaviours, and sensor payloads. Multiple input and output options should be available to the processors and should be relatively low in power consumption and be lightweight. The mission processor should have a source of power and data links for communication to the ground. The mission processor will also connect to the sensor payload directly, and this will allow the behaviours to incorporate the perception data. The perception data can be processed directly on the mission computer itself. Alternatively, it can be processed on a separate payload computer. This design allows for future sensor payloads to be easily added to the system and consumed by the autonomous behaviours. Software architecture is also required for automated systems and is discussed in the next section.

3.3.3 UAS automation software architecture

The primary purpose of the mission processor is to execute high-level mission planning and behaviour processes and to send and receive messages between processes, other vehicles, and the ground station (Tisdale *et al.*, 2006). One of the key design goals of the system is that it should be flexible with the architecture and should have the potential for expansions (Pippin, 2015). Different autonomous behaviours, sensors, and communication capabilities may be added in the future. The functionality may even run across multiple processors on separate payload computers. The software components must communicate using open and flexible architectures. An example of an open and flexible architecture is the use of message-oriented middleware. The separate processes send messages to each other, and processes have little or no knowledge of each other.

In a UAS, several processes encapsulate the details of interacting with various sensors, the autopilot system, data links, and autonomous behaviours. Each process can send or receive messages from the others. Since the processes have little or no knowledge of each other, it is easy to add processes in the future without changing the existing system. The following are examples of processes, and the message types used:

- The **avionics process** interacts with the flight control computer and obtains messages from the autopilot and avionics components related to the aircraft status. The process can send messages which include information about the position and orientation of the aircraft. It can also subscribe to waypoints or other command messages from the autonomous behaviours. In other words, these messages can check for accuracy and forward the information to the flight control computer.
- **Sensory processing** uses multiple sensors that can be operated at once to process perceptions from cameras, radar, or other sensors, onboard the aircraft. These sensors can also send messages to a related sensor. For example, a process might send a message to another sensor to indicate that an object was detected at a given location.
- **Behaviours** are the highest level of autonomous processing. It operates as a set of behaviours which can perform high-level autonomous operations and planning. For example, a behaviour focussing on searching might generate a pattern of waypoint messages that could be consumed by the avionics process and then sent to the flight controller. The behaviours can get messages from the sensor processes and change behaviour, based on the information received. The behaviours also communicate mission control information for other processes to consume.
- **Communications** are used to encapsulate access to the mission data link and do access control between the physical data link and onboard processes.
- The **filter** process could attach to any of the messages, such as sensor data and perform additional filtering or data fusion and produce the result to other processes.
- The **Joint Architecture for Unmanned Systems (JAUS)** or other standard-related processes can be used to convert messages among standard formats to support interoperability.

This approach to automation software supports extensibility and encapsulates lower-level components from other processes. Additional sensors could easily be added to the system in the future. A new standard for processes could also be added to support the translation from one standard to another for further support of interoperability. There are various system and software architectures available to use when automating a UAS. It is necessary to understand the different levels of automation. Understanding the different levels of automation can be used

to determine the level of automation of a system. The different levels of automation used within a system are discussed in the next section.

3.3.4 Automation levels

The system does not have to be fully automated. It can fluctuate across a continuum of levels, from the lowest level of fully manual performance to the highest level of full automation. There are several levels of automation between these two extremes, as defined in Section 1.2 (Parasuraman *et al.*, 2000). These measures are repeated here for completeness and are expressed on a ten-point scale. On this scale, level 1 is the lowest level of automation, and level 10 is the highest level of automation as follows:

- **Level 1:** The computer does not assist, and the human is left to make all the decisions and take action.
- **Level 2:** The computer gives a complete set of decisions, actions, and alternatives.
- **Level 3:** The computer narrows the selections down to a few.
- **Level 4:** The computer suggests one alternative.
- **Level 5:** The computer executes a suggestion only if the human approves.
- **Level 6:** Allows the human a certain amount of time to veto before performing an autonomous action.
- **Level 7:** Does autonomous actions and then informs the human of the actions.
- **Level 8:** Provides information to the human only if required.
- **Level 9:** Informs the human only if the computer decides it needs to.
- **Level 10:** The computer does everything autonomously and ignores the human.

It can be seen at level 2 that several options are given to the human by the computer, but the computer has no further say in which action is chosen. At level 4, the computer suggests one action to take, but in the end, the human still retains authority over what action should be chosen. At level 6, the computer gives the human only a certain amount of time to choose an action before acting.

Autonomous systems can operate within a specific level on this continuum. For example, a conflict detection and resolution system can notify an air-traffic controller of a conflict within the current flight paths of two or more aircraft. The autonomous system can then suggest conflict resolution. The level of automation would be categorised as level 4 automation. At level 6 or higher, the system would automatically execute its conflict resolution advisory, unless the

controller intervened. The following four levels represent the level of UAS automation (Clarke, 2014):

- **Level 1:** All high-level piloting is controlled manually.
- **Level 2:** Some high-level piloting is still controlled manually unless the piloting is switched over to automatic.
- **Level 3:** The UAS control is by default automatic, but can be switched to manual piloting.
- **Level 4:** A fully autonomous system.

Level 1 is the lowest level of automation, and level 4 is the highest level of automation. To improve the level of automation of a UAS in this study, various algorithms and tools were identified that could be used to simplify and improve UAS automated control. In the next section is a discussion on these algorithms and tools that were used.

3.3.5 Automation with programming agents

"An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators" (Russell & Norvig, 2010). The definition of agents covers an extensive spectrum. An agent can be a simple thermostat which does not learn anything new. It can also be worms that learn a small repertoire of behaviours or humans who are considered to be the most exceptional learners on earth thus far (Mills & Stufflebeam, 2005). An agent perceives its environment with the help of sensors. Something as simple as a keyboard and video camera can function as sensors when attached to the agent. An agent uses effectors as instruments to show responses from the environment that the agent perceives. Examples of effectors are monitors, printers, robots and robotic arms.

The environment of the agent is generally represented by the domain or world of the agent. The domain of the agent must be limited to a specific type of situation. If the domain of the agent is not defined at the start, an infinite number of possibilities is available to the agent. Two types of environments impact the computational challenges of the agent's program (Mills & Stufflebeam, 2005). In the first environment, the agent is non-deterministic. The environment should be easily accessible to the agent to enable the agent to effectively use its sensors to obtain complete information about the state of affairs. This information is then used by the agent to complete its goal. For example, a thermostat always has complete access to the temperature of the room, and the agent does not need to store any information. The agent has access to whatever knowledge is needed at any given time, and there is no need to store the state of the world internally. In the second environment, the environment is deterministic. The future state of affairs is deducible from the current state of affairs, and nothing is left to chance. Board games

have this feature, even though the tree of possibilities may extend out into billions of possible moves and counter-moves. There are various types of agent. Each agent is designed and used for a specific goal in mind. Next, a discussion on a simple reflex agent is provided.

3.3.5.1 Simple reflex agent

The most basic type of agent is a simple reflex agent (Mills & Stufflebeam, 2005). This type of agent selects its actions, based on its current percept, and ignores the rest of the percept history. The simple reflex agent uses a set of rules to determine what action to do next. For example, if an agent is programmed to search for a specific rock at a specific location, it will collect that rock, and if it finds the same rock at a different place, it will also pick it up. The agent does not take into account that it has already picked up that specific rock. A simple reflex agent acts upon a specified percept with one of the pre-programmed responses. Even if there are several possible responses to a single percept, the agent has a list of situation-action rules to execute for this specific percept. A situation-action rule can be explained as a simple hypothetical imperative. For example, if situation A is the current state of the agent and goal C requires plan B to be executed, then plan B is executed. Alternatively, even more simply, given A, execute B. Simple reflex agents are admired for being simple, but they have limited intelligence. The agent is useful when a quick automated response is needed. Humans have a similar automated response to fire. The human brain instructs our hand to be pulled away without even thinking that there could be other possible dangers that can be in the path of our arm. These actions are called reflex actions. Simple reflex agents are easy to work with, but their intelligence limits the complexity of the task that can be performed by the agent. Because simple reflex agents have basic intelligence, it can limit the level of automation that can be achieved by this specific agent. It is, therefore, advisable to investigate what other types of agent can be used to acquire a higher level of automation. The utility-based agent is discussed in the next section.

3.3.5.2 Utility-based agent

In more sophisticated agents, such as a utility-based agent, a utility measure is applied to the different possible actions that can be performed in the specific environment (Mills & Stufflebeam, 2005). The utility-based agent will try and produce the best outcome by rating each scenario to determine if it will achieve specific criteria. A utility-based agent will consider the following: the probability to succeed, the number of resources that are needed to complete the scenario, the importance of the goal that needs to be achieved, and the time it will take to complete the scenario. These considerations are all factored when the utility function is calculated. The behaviour of the agent is not only built on its knowledge about the world but also the agent's percept sequences. The sequences of perceptions are not always predictable

by the agent. This unpredictability is found mainly in non-deterministic environments. The agent obtains input from its sensors throughout the constant changing world. The agent can decide to map its percept sequences to formulate a plan of action in pursuit of its goals that need to be completed. The architecture of the agent can also include a learning program so that the agent can go through trial and error in a novel situation to accomplish its goals.

It is difficult to predict every state of the world in which the agent will be, and writing rules for each scenario would be laborious. The solution is to give the agent some goals, the ability to re-evaluate its state continuously, the ability to learn through a trial and error process and ways to evaluate possible plans that can become a possible path to get to the goal. Artificial neural networks can be used to aid the utility-based agent employed in this study to learn. Next, a discussion on artificial neural networks is provided.

3.3.6 Artificial neural networks

Artificial neural networks are statistical models that are inspired by the biological neural network of the brain (Castrounis, 2016). Artificial neural networks are capable of modelling and processing nonlinear relationships between inputs and outputs in parallel (Ranasinghe *et al.*, 2017). The algorithms that are used with these neural networks are part of machine learning and can be used in several applications (Deng & Yu, 2013).

The character of the neural network is defined by the adaptive weights and the paths between the neurons. The weights and paths can be adjusted by a learning algorithm which learns from data. It enables the model of the neural network to be improved. Neural networks should also be given an appropriate cost function. Cost functions are used by the neural networks to learn the best solution for the problem that needs to be solved (Castrounis, 2016). The process involves determining the best values for all the model's parameters. It includes the neuron path adaptive weights that are the primary target and the algorithm's alteration of parameters such as the learning rate. The implementation requires optimisation techniques, such as stochastic gradient descent or gradient descent. The optimisation techniques need to try and optimise the solution of the neural network. Solutions need to be as close as possible to the optimal solution. If the optimisation technique is successful, it means that the neural network is capable of solving the intended problem with a higher performance (better accuracy rate in this study).

The architecture of an artificial neural network is modelled by layers of artificial neurons, also known as computational units. These units can receive input and apply an activation function with a threshold to determine if the message passes through the network (Suzuki, 2011). The most basic neural network model consists of an input layer, followed by one or more hidden layers, and lastly, an output layer. Neural network models can become more complicated by increasing the number of hidden layers and the number of neurons in any of the layers or the

number of paths that are between neurons. As the model becomes more complex, it also increases abstraction and problem-solving capabilities. It should be noted that overfitting (when the performance on a test set is much lower than the performance on the training set) can occur when the complexity of the model is increased too much.

The primary components of neural network techniques are the architecture and adjustment of the model, in addition to the actual learning algorithms themselves (Castrounis, 2016). These characteristics of a neural network can have a substantial impact on how the model performs. The models are also characterised and adjustable by the activation functions that are used to convert a neuron's weighted input into its output activation. Several different transformations can be used as the activation function. The abstraction of the output is a form of distributed representation as a result of the transformation of the input data that goes through neurons and layers and is in contrast to a local representation. An example of local representation is the means that is represented by a single neuron. An entire network is a form of distributed representation due to the many transformations across layers and neurons. Artificial neural networks are extremely powerful, can be very complicated, and are considered to be black box algorithms, as the inner working of the neural network is sometimes challenging to understand and explain. Various artificial neural network algorithms can be used for specific goals in mind. The radial basis function neural network architecture utilised in the study is discussed in the next section.

3.3.6.1 Radial basis function neural networks

The radial basis function neural network (RBFNN) was introduced by Broomhead & Lowe (1988). They introduced the RBFNN for function approximation, time-series forecasting, and classification or clustering tasks. RBFNNs have been extensively researched and studied over recent years (Pazouki *et al.*, 2015). RBFNNs have been successfully used in many applications, including interpolation, chaotic time-series modelling, speech recognition, image restoration, three-dimensional object modelling and data fusion. Figure 3-5 is a visual representation of the architecture of an RBFNN. The RBFNN architecture consists of an input vector, a layer of RBF neurons, and an output layer per category or class of data. The input vector is the *n*-dimensional vector that needs to be classified. Each RBF neuron knows the entire input vector. RBFNNs act on the input patterns and send the outcomes to the output neurons, which are in the output layer. The output neuron of the network is the weighted sum of the patterns of the hidden neurons.

Generally, an RBF is a multivariate function $\Phi: \mathbb{R}^s \rightarrow \mathbb{R}$, such that

$$\Phi(x, x^c) = \phi(\| x - x^c \|), \tag{3-1}$$

where $\phi: [0, \infty) \rightarrow \mathbb{R}$ is a univariate function, x is the input, x^c is the centre point of the RBF, the norm $\|.\|$ is typically the Euclidean distance, and s is the dimension of input patterns. The centre points (μ_k in Figure 3-5) and shape parameter (r in Equation 3-2) characterise the RBF.

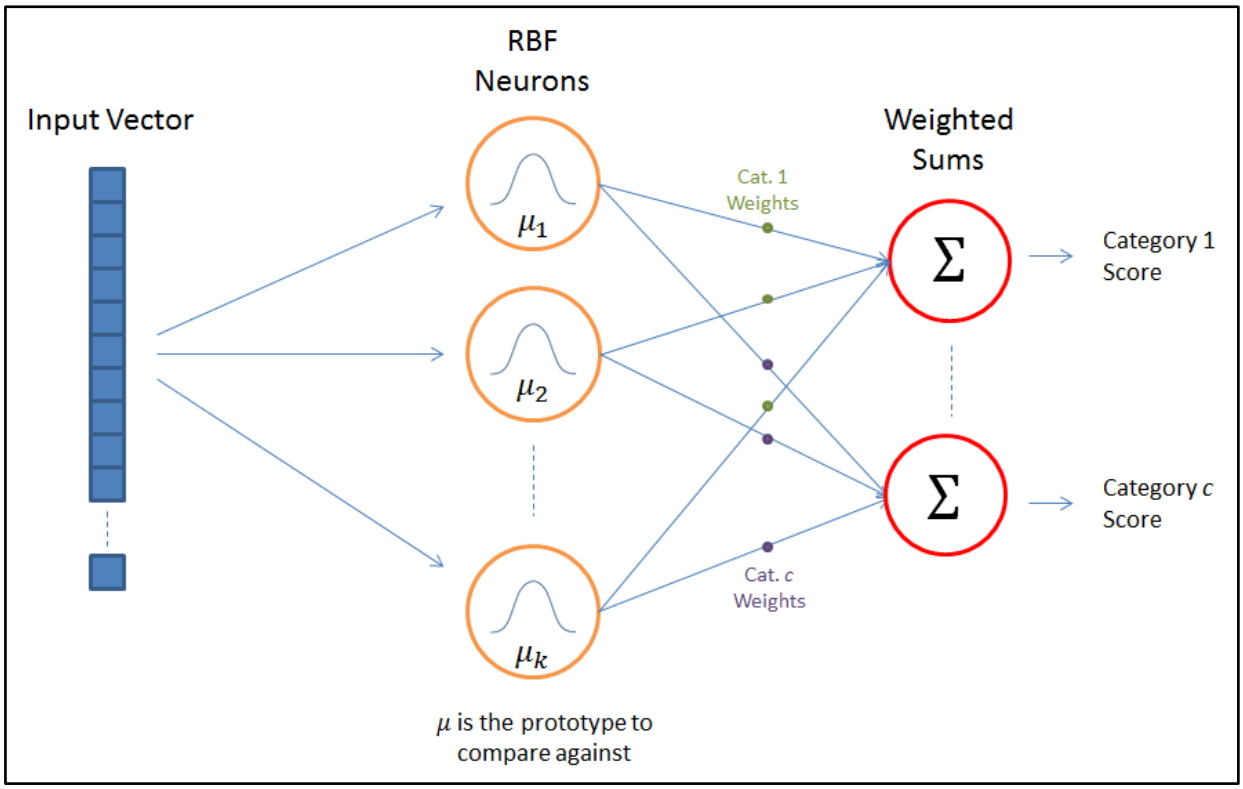


Figure 3-5: Radial basis function neural network (Pazouki *et al.*, 2015)

An RBF reaches its maximum value when it is applied to the centre points and decline when applied to points far away from the centre points. The decrements are affected by the shape parameter of the RBF, as seen in Figure 3-6.

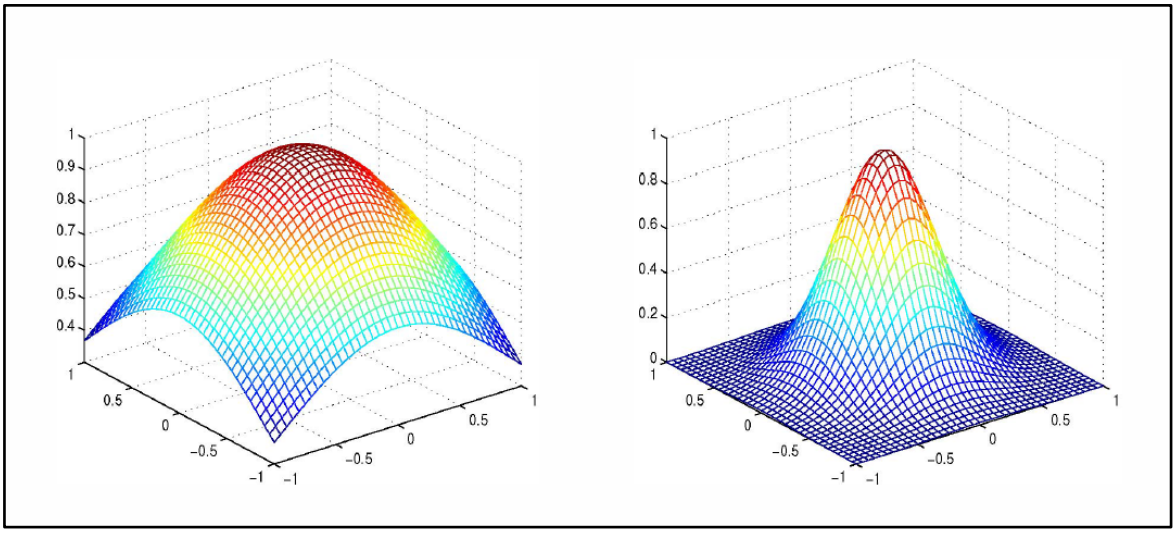


Figure 3-6: Gaussian with r = 1 on the left and r = 1/3 on the right that is centred at the origin

The Gaussian function is often used as the RBF, i.e.

$$\phi(\varpi) = \exp(-r^2\varpi^2), \tag{3-2}$$

where $\varpi \in \mathbb{R}$ and $r > 0$, known as the *radius* is the shape parameter. The Gaussian function Equation 3-2 can be expressed explicitly as the following RBF in Equation 3-1:

$$\Phi(x, x_i^c) = \exp(-r^2 \parallel x - x_i^c \parallel^2) = \exp(-\parallel x - x_i^c \parallel^2/2\sigma^2), \tag{3-3}$$

where $\sigma^2 = 2/r^2$ is the variance of the normal distribution.

For a specified input pattern $x \in \mathbb{R}^s$, the typical output $o(x) \in \mathbb{R}$ of an RBFNN is

$$o(x) = \sum_{i=0}^{n} w_i \Phi(x, x_i^c), \tag{3-4}$$

where *n* is the number of neurons in the hidden layer, w_i is the weight of hidden neuron i and x_i^c is the centre point of hidden neuron i.

An RBFNN is opportune for function approximation and pattern recognition due to the simple topological structure of the RBFNN and the ability of the RBFNN to reveal how learning proceeds in an explicit manner (Tao, 1993). In the next section is an example of solving a simple XOR classification problem by using an RBFNN.

Example of RBFNN implementation

Figure 3-7 denotes a two-dimensional vector having two values, namely X_1 and X_2. On the X_1 axis, X_1 can have values 0 and 1. On the X_2 axis, X_2 can also have values 0 and 1.

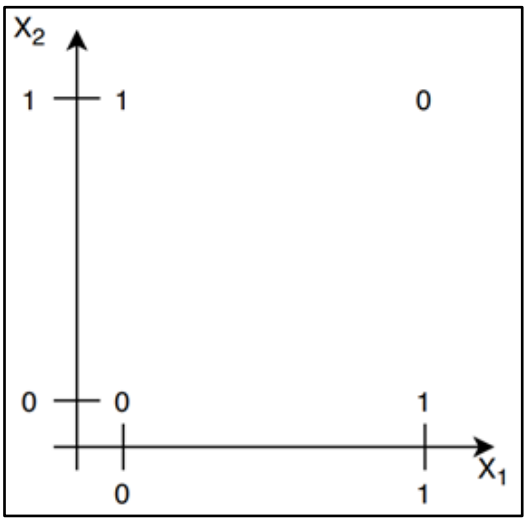


Figure 3-7: Visual representation of XOR classification problem

The XOR function is defined as follows in Table 3-1:

X_1	X_2	XOR function
0	0	0
0	1	1
1	0	1
1	1	0

Table 3-1: XOR function.

This two-dimensional vector is then cast into a four-dimensional vector by using four Radial basis functions, as shown below:

$$\phi_1 \rightarrow t_1 = (0,0);\ \sigma_1 = 1;\ \phi_1(x) = \exp\frac{-\|x-t_1\|^2}{2} \quad (3\text{-}5)$$

$$\phi_2 \rightarrow t_2 = (0,1);\ \sigma_2 = 1;\ \phi_2(x) = \exp\frac{-\|x-t_2\|^2}{2} \quad (3\text{-}6)$$

$$\phi_3 \rightarrow t_3 = (1,0);\ \sigma_3 = 1;\ \phi_3(x) = \exp\frac{-\|x-t_3\|^2}{2} \quad (3\text{-}7)$$

$$\phi_4 \rightarrow t_4 = (1,1);\ \sigma_4 = 1;\ \phi_4(x) = \exp\frac{-\|x-t_4\|^2}{2} \quad (3\text{-}8)$$

In the first part of the equation, are four radial basis functions, namely $t_1 \ldots t_4$. In the second part, is the spread for the radial basis functions, $\sigma_1 \ldots \sigma_4$. In this example, the value of 2 was chosen for *P* (number of nearest receptors). This means that the two nearest neighbours need to be chosen. For every receptor given, there are three neighbours, two of the neighbours are at a distance of 1, and one is at a distance of 1.4 $(\sqrt{2})$. Table 3-2 is a summary of the calculations. The four feature vectors can be seen in the input column, followed by the function values for each RBF function. In the last row are the weights that were chosen and used to calculate the values in the output column. The output was calculated as follows: if the value (in the second last column) is more than 0 the output is 1 and if the value is less than 0 the output is 0.

Input	$\emptyset_1$	$\emptyset_2$	$\emptyset_3$	$\emptyset_4$	$\sum w_i\emptyset_i$	Output
(**0, 0**)	1.0	0.6	0.6	0.4	-0.2	0
(**0, 1**)	0.6	1.0	0.4	0.6	0.2	1
(**1, 0**)	0.3	0.4	1.0	0.6	0.2	1
(**1, 1**)	0.4	0.6	0.6	1.0	-0.2	0
	-1	+1	+1	-1		

Table 3-2: Summary of calculations

In the next section, an overview of some of the advantages of using an RBFNN are presented.

Advantages of an RBFNN

RBFNNs have the advantages of easy design, good generalisation, strong tolerance to input noise, and online learning ability (Yu *et al.*, 2011). The favourable properties of an RBFNN make it suitable to design flexible control systems. The RBFNN also has a quick turnaround time for processing training data (Zemouri *et al.*, 2003). The RBFNN is widely used for pattern recognition tasks for its advantage of having fast learning algorithms (Ghosh & Ari, 2011). One of the proposed solutions in this study is to use a smartwatch with an RBFNN to simplify and automate UAS control. The data gathered from the smartwatch needs to be pre-processed for the RBFNN. In the next section is a discussion of the Fast Fourier Transform algorithm that can be used to pre-process the data for the RBFNN.

3.3.7 Fast Fourier Transform algorithm

The Fourier Transform is a mathematical operation that is used to express any function with regard to time as a function in regard to frequency (Rao *et al.*, 2011). A Fourier Transform is useful for signal processing, as all signals being analysed are a function of time. By analysing a single function concerning frequency rather than time can give much information about the source, whether it is a broad sound, a specific colour, or a radio signal at a specific frequency.

A Discrete Fourier Transform (DFT) is a form of Fourier Transform that transforms a discrete-time input into a frequency. The DFT is used for any continuous signals that are sampled over a limited amount of time and transformed into a frequency. The DFT is sufficient for frequency analysis; however, performing mathematical operations can be very computationally costly.

The Fast Fourier Transform (FFT) algorithm is an algorithm that can be used to improve the complexity of the original DFT and still effectively transform a signal into its frequency domain. A general description of the algorithm is given in this section to explain the workings of the algorithm. In complex notation, the time and frequency domains each contain one signal composed of N complex points (Smith, 1997). Each of these complex points is made up of two numbers; one is the real part, and the second is the imaginary part. The FFT works by first decomposing an N point time-domain signal into N time-domain signals each composed out of a single point. Secondly, the N frequency spectra that correspond to these N time-domain signals are calculated. The final step is to synthesise the N spectra into a single frequency spectrum. General FFT algorithms are sometimes limited to data of arbitrary length. The Bluestein FFT algorithm is not restricted to data of arbitrary length. The Bluestein, also known as the chirp z-transform algorithm, is an FFT algorithm that computes the discrete Fourier transform (DFT) of arbitrary sizes by re-expressing the DFT as a convolution (Agarwal *et al.*, 1994). The FFT algorithm is used in this study to transform accelerometer data into signals that are then

processed by the RBFNN. In the next section is a discussion on techniques that can be used for automating and simplifying UAS control by using gesture and activity recognition.

3.3.8 Automating UAS control with hand-held and wearable devices

UAVs are traditionally controlled by using joystick remote controllers, mobile applications, and embedded computers (Natarajan *et al.*, 2018). Using traditional control methods might not always be natural or inherent (Lu *et al.*, 2017). It is, therefore, necessary to investigate techniques that could be used to interact with the UAVs, using hand-held and wearable devices.

Although this study focuses on gesture and activity recognition with hand-held and wearable devices, it is also useful to investigate vision-based gesture and activity recognition and why it might not be sufficient for UAV control. Vision-based gesture recognition UAV control is discussed in the next section.

3.3.8.1 Gesture recognition with a UAV camera

Natarajan *et al.* (2018) developed a vision-based hand gesture recognition framework for UAV control. In this implementation, a video stream is continuously recorded and then segmented into a sequence of still images. These images are then analysed through a hand gesture recognition process. The process includes feature extraction, hand region identification, and gesture classification. The gesture detected is then mapped to various UAV control instructions. Five gestures were used for this implementation and included the following: fist, palm, go symbol, v-shape, and little finger. Classification results showed that an average accuracy of 90 percent was obtained when the UAV was within three feet of the pilot's hand. When the distance between the pilot and the UAV was increased, the average accuracy decreased. One advantage of the vision-based hand gesture system is that most commercialised UAVs already have a camera onboard. Using vision-based gesture recognition is usually also limited by the intensity of light and the surrounding environment (Vu *et al.*, 2012). One possible solution to obtain higher accuracy is to use a higher quality camera on the UAV. The alternative is to use hand-held and wearable devices for gesture recognition; this is discussed in the next section.

3.3.8.2 Gesture recognition using hand-held and wearable devices

Vu *et al.* (2012) implemented a hand gesture classification system that used an armband with one inertial measurement unit (IMU) and three mechanomyography (MMG) sensors to gather acceleration data. The following gestures were used: clenching fist, snapping, thumb up, thumb down, double tap on the forearm, palm up, palm down, and a double clenching fist. These gestures were mapped to various high-level UAV controls. A convolutional neural network was used for gesture classification. The results showed that five out of eight gestures were classified

with an average of 94.38 percent accuracy, with an overall average of 85.4 percent accuracy. The experiment showed a sufficient accuracy rate, but based on the implementation, various sensors are required, and the armband is rather bulky, as shown in Figure 3-8, compared to a smartwatch.

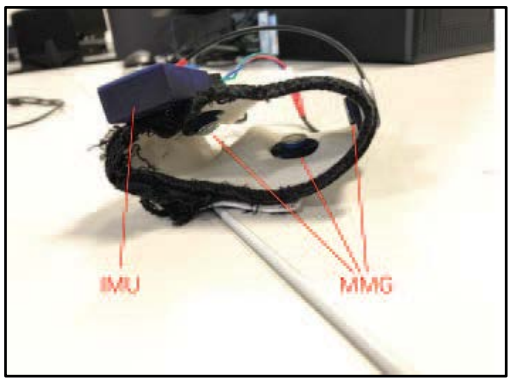


Figure 3-8: Armband with one IMU and three MMGs

Liu *et al.* (2020) implemented a wearable human-machine interface based on MMG signals. A three-axis accelerometer was fixed to the strap of a smartwatch to measure MMG signals generated by the extensor digitorum muscle (muscle on the forearm). An Arduino was used to enable communication between the sensors and a computer. The system was used to capture eight different gaming gestures, namely finger-snapping, index finger flick, fist, clapping, coin flip, wrist extension, wrist flex, and shooting. These gestures were used with the *k*-nearest neighbour's algorithm for classification. The results showed an average accuracy between 85 percent and 98 percent. In this experiment, specialised sensors were used to enable gestures to be recorded based on the movements made by a person's forearm. The experimental results were not used to control a UAV, but the classifications could be mapped to UAV control instructions. Vision-based activity recognition UAV control is discussed in the next section.

3.3.8.3 Activity recognition with a UAV camera

Mliki *et al.* (2020) developed a new approach for human activity recognition by using video sequences captured from a UAV camera. The approach entails two phases: an offline phase and an inference phase. In the offline phase, the human activity recognition model is created with a convolutional neural network. Scene stabilisation is also done in the offline phase based on the calculation of the optical flow applied to detect the possible motions in the scene recorded. In the inference phase, the trained model is used to detect humans and recognise their activities. Two different methods carry out human activity classification. The first method is to classify video frames, and the second method is to classify an entire video sequence. The following activities were identified for classification: walking, running, digging, throwing, and waving. The overall results showed an accuracy of 56 percent for classification on video frames

and 68 percent accuracy on sequence classification. The accuracy of vision-based activity recognition is relatively low when compared to activity recognition with hand-held and wearable devices which is discussed in the next section.

3.3.8.4 Activity recognition using hand-held and wearable devices

Smartwatches and smartphones have accelerometers embedded that can sense a user's movements and can help identify the activity the user is performing. Weiss *et al.* (2016) compared smartwatch- and smartphone-based activity recognition accuracy using standard machine learning algorithms. Accelerometer data were recorded for various activities (walking, running, standing, sitting, eating, and writing). The smartwatch accelerometer data provides an accuracy of 91.9 percent for activity recognition, and the smartphone accelerometer data had an accuracy of 72.6 percent. Based on these results, it is better to use a smartwatch for activity recognition.

Lu *et al.* (2017) proposed a wearable approach to control a UAV. The system used inertial sensors embedded inside a smartwatch and smartglasses. The smartwatch is used to sense the periodic movement of the pilot's arm for step detection. Smartglasses are used to sense the movement of the pilot's head with the use of an accelerometer and then to move the UAV, based on the pilot's head movements. A hand-held device is used to link the smartwatch and smartglasses and send pitch, roll, and yaw commands to the UAV. Motion recognition was designed to enable the UAV to take-off/rise, descend, stop rising or descending, and take photos based on the accelerometer data gathered from the smartwatch. The experimental results showed that the accuracy of step detection is above 92 percent. The disadvantage of this implementation is that the UAV moves when the pilot moves his/her head, which might not always be the desired behaviour.

Based on the discussions above, there are several techniques available for gesture and activity recognition to enable simplified UAS control and to improve automation. The tools and platforms that are used for implementing a possible solution to improve automated UAS control systems are discussed next.

3.4 Proposed research design

This section is divided into two parts. The first part focuses on the hardware that is used in this study, followed by a discussion on the software that is used.

3.4.1 Hardware

Two different types of Parrot quadcopter, the Parrot AR.Drone and Parrot Bebop, were made available for this study. Next, an overview of these quadcopters is given to identify which quadcopter is better suited for the system implementation.

3.4.1.1 Parrot AR.Drone

The Parrot AR.Drone 2.0, as seen in Figure 3-9, is a high-tech flying toy, and it can also be used for augmented-reality games (Bartak & Vykovsky, 2016). On the technical side, it is a quadcopter with sensors and a controller. The AR.Drone has two cameras; one of the cameras is facing down and is used by the onboard software to stabilise the drone by preventing drift (François *et al.*, 2011). The front camera is used for video streaming or to take photos, has a resolution of 1280 x 720 pixels and runs at 30 fps (Bartak & Vykovsky, 2016). The AR.Drone is equipped with a three-axis gyroscope measuring pitch, roll, and yaw. It also has a magnetometer that increases the accuracy of these measurements (François *et al.*, 2011). The AR.Drone has a 3-axis accelerometer that can be used to measure acceleration in all three dimensions. A pressure and ultrasound sensor measures the altitude of the AR.Drone. Altitude is measured just above the ground by the ultrasound sensor and is mostly used during take-off and landing (Bartak & Vykovsky, 2016). The pressure sensor measures the altitude a few feet above the ground when the ultrasound sensor does not give a reasonable estimate.

The AR.Drone is powered by a 1GHz 32bit ARM Cortex A8 processor with 1GB DDR2 RAM that runs at 200MHz (Bartak & Vykovsky, 2016). The operating system of the AR.Drone is GNU/Linux, which makes it possible to install software on it, but it has limited power, so alternatively, an external computer can be used. The motherboard embeds the processor, a Wi-Fi chip, the vertically-oriented camera and a connector to the front camera (François *et al.*, 2011). The AR.Drone is controlled externally via WLAN through a set of commands which is sent from the ground to the quadcopter. Three UDP (User Diagram Protocol) channels are used for all communication. The command channel is used to send commands to the AR.Drone. Attention commands sent through UDP port 5556 are used to control and configure the drone. These commands are sent 30 times per second (Piskorski *et al.*, 2012). The following commands can be sent to the AR.Drone at a frequency of 30Hz: take-off, land, setting limits, calibrate sensors, swap cameras, set rotation speed of rotors, set pitch, roll, yaw, and vertical speed (François *et al.*, 2011).

The state of the AR.Drone can be obtained by using the navigation data (Navdata) channel which sends Navdata from the drone to the client via UDP port 5554 and provides the following data: state of the drone (flying, steady, landing, take-off, calibration, booting), and sensor data (current pitch, roll, yaw, altitude, battery remaining, and the speed of all rotor axes). The

navigation data also includes tag detection, which can be used to create augmented reality games. The last channel is the stream channel, and it provides visual information either from the front view or the bottom view (Bart and Vyˇ, 2015). Views can be switched by merely sending a particular command to the AR.Drone indicating which view needs to be activated. The video stream is sent from the AR.Drone to the client on UDP port 5555. These images and video stream can be decoded using a codec which is included in the SDK (Piskorski *et al.*, 2012).

Figure 3-9: Parrot AR.Drone 2.0

The complexity of the Parrot AR.Drone is abstracted to only four intuitive inputs (pitch, roll, yaw, and vertical velocity). This enables the researchers to focus on the particular field of research without being concerned about problems such as sensor fusion and the optimal flow of algorithms. The Parrot Bebop Drone is discussed next.

3.4.1.2 Parrot Bebop Drone

The Parrot Bebop Drone can be controlled by using the Freeflight 3 (a very intuitive application). Freeflight 3 is developed for iOS and Android Smartphones and tablets (Parrot, 2014). The application is user-friendly, allowing the pilot to focus on flying the drone. The return to home function returns the Bebop Drone directly to its starting position when the altitude is greater than 10 metres. If the altitude is less or equal to 10 metres, the Bebop Drone will rise and stabilise itself at 10 metres before returning to the original take-off position in a straight line. When it

reaches the take-off position, it will stop and hover two metres above the ground. A deconstructed picture of the Bebop Drone can be seen in Figure 3-10.

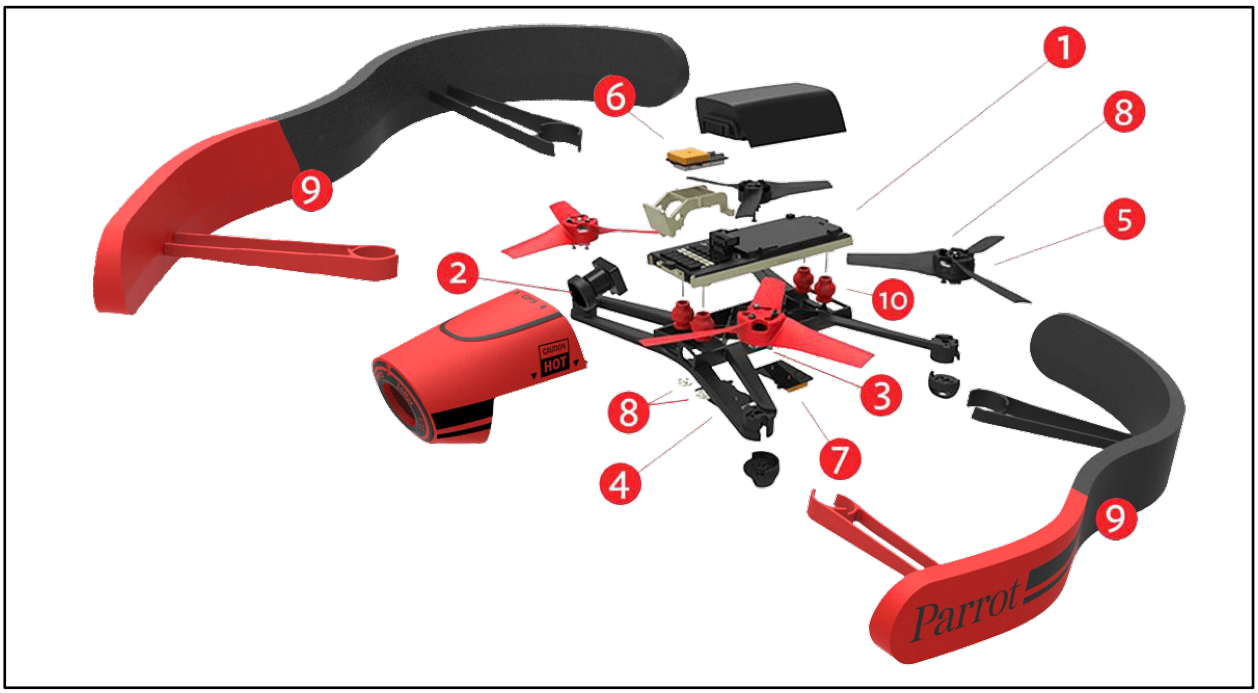


Figure 3-10: Parrot Bebop architecture

The drone has the following components:

1. Motherboard – The navigation computer features a Parrot P7 dual-core CPU and quad-core GPU with 8GB flash memory. These items are fixed on a magnesium shelf that acts as electromagnetic shielding and as a radiator. The operating system is based on Linux, and an open-source SDK is available for development. The Parrot Bebop Drone computer is eight times more powerful than the Parrot AR.Drone 2.0's onboard computer.
2. Camera – The Parrot Bebop has a 14-megapixel "Fisheye" camera. The drone records video and pictures in a 180° field with good image quality. The drone has full-digital image stabilisation technology that allows the Bebop Drone to take stable and clear aerial footage when the drone moves. The lens is splash and dustproof.
3. The four brushless out-runner motors
4. The glass fibre reinforced ABS structure (weighing 400g) makes the Parrot Bebop Drone robust and safe. If a collision occurs, the propellers will automatically stop. The emergency enables the drone to land immediately.
5. Three-blade auto-block propellers in polycarbonate with fast disassembly system

6. Inertial measurement unit – The Parrot Bebop has a GPS that allows for the return to home function to quickly bring the Bebop drone back to its take-off point, a three-axis accelerometer, a three-axis gyroscope, a three-axis magnetometer, and a pressure sensor. Data analysed from these sensors ensures optimal stability without compromising the manoeuvrability.
7. Wi-Fi MIMO (multiple input, multiple output) antennas – The Bebop comes with two double-set ceramic antennas that allow it to handle both 2.4 and 5 GHz frequencies. It generates its own Wi-Fi 802.11 network. Depending on the network interface, the user can select the frequency of his/her choice.
8. (a) Optical-flow sensor – The Bebop Drone has a vertical stabilisation camera that takes an image of the ground every 16 milliseconds and compares it to the previous reading to determine the speed of the drone.

   (b) Ultrasound sensor – Analyses the flight altitude up to eight metres.

   Both of these sensors are used to track the speed of the drone.
9. High-resistance expanded polypropylene outdoor hull – It clips and unclips easily to adapt to indoor and outdoor flight and protects the propellers against potential bumps. It can be removed to reduce wind factors.
10. Anti-vibration bumpers

3.4.1.3 Comparing Parrot Bebop and AR 2.0 Drones

The SDKs for the Parrot AR 2.0 and Bebop are freely available on the Internet (Parrot, 2012). The SDK contains several libraries that can be used to develop applications and have functionalities such as video streaming, joystick and hand-held controller input, navigation data acquisition, and display (Martin, 2012). In Table 3-3 is a comparison between the Bebop and AR 2.0 drones. These two drones are very similar, but the Bebop has some advantages over the AR 2.0 drone. One of the significant advantages is the quality of the camera. The Bebop has a 14-MP camera, and the AR 2.0 has a 0.9-MP camera. The signal range of the Bebop is 200m further than the AR 2.0, meaning that the drone can be flown further from the control system. The battery life of the Bebop is almost double the AR 2.0 battery life. This means that the Bebop can endure longer flight times. One major disadvantage of the AR 2.0 is that it is an older model of the Parrot drone range and it is not available for purchase from the Parrot store anymore. It is because of these advantages that the Bebop Drone is chosen to conduct the experiments. The smartwatch that is used for capturing gestures is described in the next section. A more detailed discussion on how the smartwatch is used for simplifying UAS control is given in Chapter 4.

Description	BeBop	AR 2.0
Camera		
Resolution	14 MP	0.9 MP
Maximum video resolution	1080p	720p
Video Stabilizer	Yes	No
Dimension (with indoor protection)		
Height	380mm	584mm
Width	330mm	584mm
Depth	36mm	127mm
Weight	410g	1810g
Connectivity		
Wi-Fi	802.11 b/g/n/ac	802.11 b/g/n/ac
Signal range	250m	50m
Battery		
Battery life	22 min.	12 min.
Battery capacity	1,200 mAh x 2	1,000 mAh
Sensors		
Accelerometer	Yes	Yes
Gyroscope	Yes	Yes
Barometer	Yes	Yes
Magnetometer	Yes	Yes
GPS	Yes	Yes
Availability		
Still sold	Yes	No

Table 3-3: Comparison between Bebop and AR.2.0 Drone (Parrot, 2018)

3.4.1.4 Moto 360 smartwatch

The wearable device that is available for this research study is the Moto 360 (2nd generation). According to reviews by Vergara (2015) and Charara (2015), the specifications of the Moto 360 (2nd generation) smartwatch are summarised below in Table 3-4. The essential sensor used on the smartwatch for the experiments is the accelerometer. The accelerometer used in a smartwatch is an electromechanical device which is employed to measure acceleration forces (Weiss *et al.*, 2016). These forces can be static, like the constant force of gravity, or they can be dynamic and caused by movement or vibration. Accelerometers can measure acceleration on one, two, or three axes. Accelerometers contain capacitive plates internally, and some of these are fixed, while others are attached to minuscule springs that move internally as acceleration forces act upon these sensors. When the plates move, the capacitance between them changes. The changes in the capacitance are used to determine the acceleration. The Moto 360 (2nd Generation) smartwatch has a nine-axis motion processing chip that combines a three-axes gyroscope, three-axes accelerometer, and three-axes compass.

Specification	Description
Display	IPS LCD screen and features a 360 x 330 resolution which is protected by a Corning Gorilla Glass 3 panel
Performance	Qualcomm Snapdragon 400 MHz processor and 512 MB of RAM
Hardware	The watch comes with the same type of heart rate monitor that is available on almost every Android Wear smartwatch. The Moto 360 uses a wireless charging dock and has a 400 mAh battery. The watch has 4GB of internal storage.
Sensors	Accelerometer, ambient light sensor, gyroscope, and vibration/haptics engine
Software	Android Wear

Table 3-4: Moto 360 (2nd generation) specifications

Next, a discussion on the software platform that is used for this study is provided.

3.4.2 Software

Various development platforms are available to use for automating UAS control. In the next section is a short overview of the software platforms that are used for this study.

3.4.2.1 Weka

Weka is a collection of machine learning algorithms and data-processing tools that are used for data-mining tasks (Frank *et al.*, 2016). Weka is designed so that the user can easily try out existing methods on new data sets in flexible ways. Weka provides the user with extensive support for the process of experimental data mining which includes the following: the preparation of the input data, evaluating learning schemas statistically and presenting the input data and result of learning visually. The user can pre-process a data set, feed it into a learning scheme, and analyse the classifiers' results and performance.

Weka includes the following methods which can be used for the primary data-mining tasks: regression, classification, clustering, association rules, and attribute selection. These algorithms take input from a single relational table that can be read by a file, or it can be generated by using a database query. One way to use Weka is by applying a learning method to a given data set and analysing the output to learn more about the data. A second method is to use learned models to produce forecasts on new occurrences. Another method is to apply several different learners to the problem and to compare their performances to choose one that will be used for the predictions. An overview of Android Studio is given in the next section.

3.4.2.2 Android Studio

Android Studio is the official Integrated Development Environment (IDE) used for Android application development, based on IntelliJ IDEA (Android Studio, 2017). Android Studio

provides the fastest tools that are required to build applications on every type of Android device. An overview of MATLAB™ follows.

3.4.2.3 MATLAB™

MATLAB™ (matrix laboratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. MATLAB™ is a proprietary programming language that is developed by MathWorks. MATLAB™ allows for matrix manipulations, plotting of functions and data, implementations of algorithms, creation of user interfaces, and interacting with programs written in different languages including C, C++, C#, Java, Fortran, and Python. MATLAB™ is intended for numerical computing, but can do symbolic computing through the use of an optional toolbox called the MATLAB™ symbolic engine. MATLAB™ will be used for the FFT (Fast Fourier Transform) calculations on accelerometer data.

3.5 Conclusion

In this chapter, the focus was on some of the available literature related to the current knowledge of this field of study. Various topics were discussed in the literature review, each aiding in the development of the solution to the problem statement that is addressed in this study. The information gathered from this chapter is used to aid in the development of possible solutions and is discussed in Chapter 4.

CHAPTER 4 SYSTEM DESIGN AND IMPLEMENTATION

In this chapter, the knowledge gained from the literature review in Chapter 3 is used for the design and implementation of proposed systems that can be used for simplifying UAS control and improving automation. The proposed systems are implemented to identify and evaluate different methods that can be used to improve and simplify the control of a UAS. Each part of the system is designed and developed by incorporating Scrum principles. Sprint, as mentioned in Chapter 2, is one of the Scrum tools that can be used for system design and implementation. Each sprint builds on the previous sprint to improve automation and simplify UAS control.

The first system is implemented in Section 4.1 with the use of basic control techniques. In Section 4.2 is a discussion on using the smartwatch accelerometer data to control the UAS. The third system which uses gesture control to improve UAS control is discussed in Section 4.3. In Section 4.4 is a discussion on the UAS follow function. Improved UAS control with automated activity recognition is provided in Section 4.5. The chapter is concluded in Section 4.6. Next is a discussion on basic UAS control.

4.1 Basic UAS control implementation

It is essential to investigate the basic UAS control system to gain knowledge of the workings of the system from a practical point of view. In Section 3.1.3.1, it is mentioned that remote control is traditionally used for UAS control. The first implementation is, therefore, implemented with the use of remote control. Figure 4-1 is a high-level visual representation of a traditional UAS control architecture.

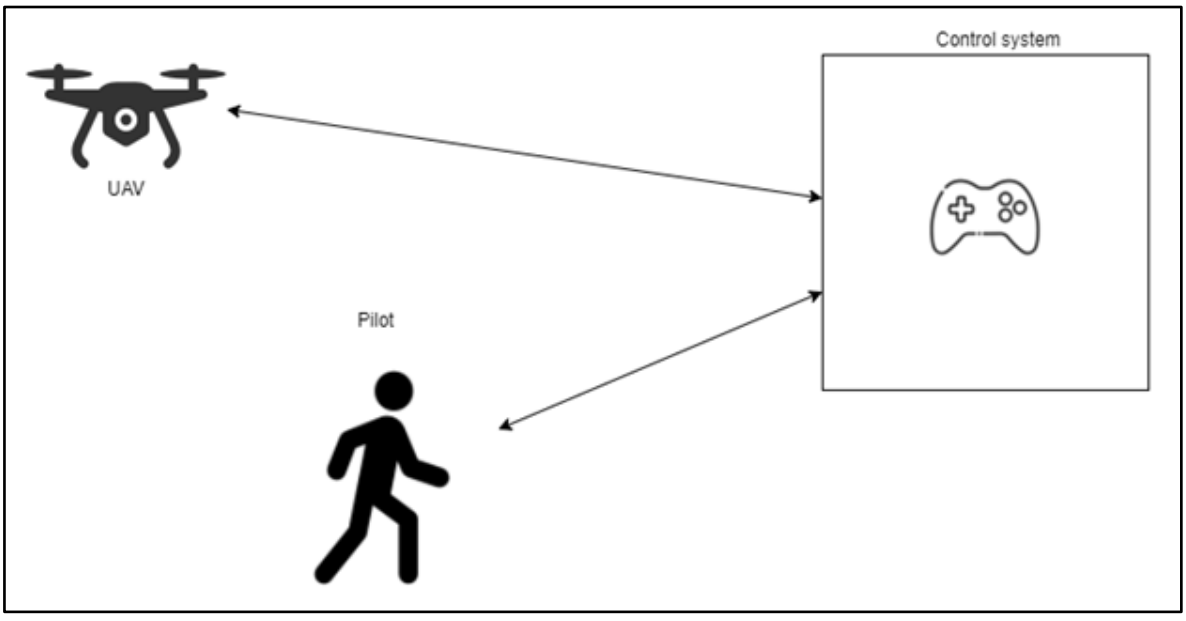


Figure 4-1: **High-level visual representation of a UAS traditional control architecture**

The traditional UAS control architecture consists of the UAV, the control system that can be any form of control that can be used by the pilot to remotely control the UAV, and the pilot who uses the control system.

The Parrot Bebop drone is used for all the experiments and shown in Figure 4-2.

Figure 4-2: Parrot Bebop drone

The creators of the Parrot Bebop drone made the software available for researchers and developers to use. Parrot also provides documentation and a sample project that can be used. The sample project uses a smartphone as a remote control. A visual representation of the smartphone user interface of the sample project can be seen in Figure 4-3. The controls are overlaid on the live image obtained from the camera mounted on the drone. The user interface is mostly used for manual control, except for the take-off, landing (indicated as a download in the figure), and emergency functions which are autonomously controlled. The sample project provides the base for building an application that can be used for improving and simplifying UAS control.

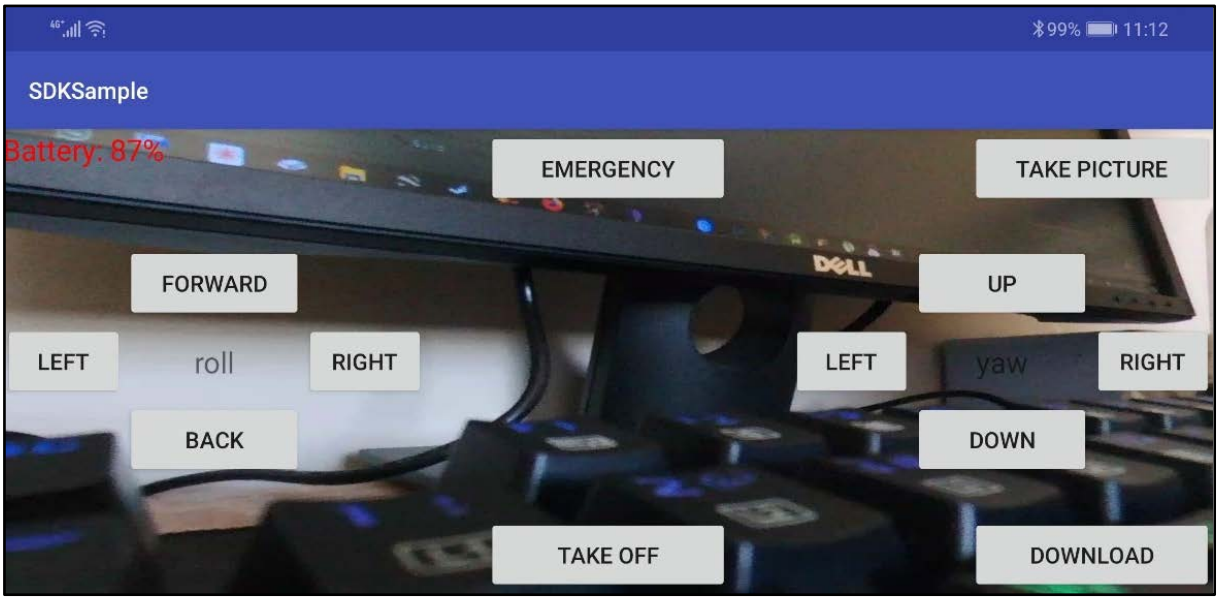


Figure 4-3: **Visual representation of the sample project's smartphone user interface**

The next section is a discussion on using the Moto 360 (2nd generation) smartwatch shown in Figure 4-4 and smartphone to simplify UAS control.

Figure 4-4: Moto 360 (2nd generation) smartwatch

4.2 Controlling the UAS with smartwatch accelerometer data

It is possible to use the smartwatch to communicate with the smartphone and then send control instructions to the UAS. This would mean that the user interface shown in Figure 4-3 needs to be replicated on the smartwatch. One problem with this implementation is that the user control interface is too big to fit on the smaller screen of the Moto 360 smartwatch. One possible solution is to make the controls smaller so that they can fit on the smartwatch screen, but this could make it difficult to press a single button and can cause multiple buttons to be pressed at once. An alternative that was explored in this study was to use the accelerometer data from the smartwatch to control the UAS directly. A high-level overview of this implementation can be seen in Figure 4-5. The Moto 360 smartwatch is small and can easily be worn on the pilot's wrist. With this implementation, instead of pressing on a button, the pilot can move his/her arm in the direction that the UAS needs to move instead of pressing on a button. The accelerometer data is gathered from the smartwatch when the pilot moves his/her arm. The accelerometer data is then sent to the smartphone, which updates the pitch, yaw or roll values of the UAS.

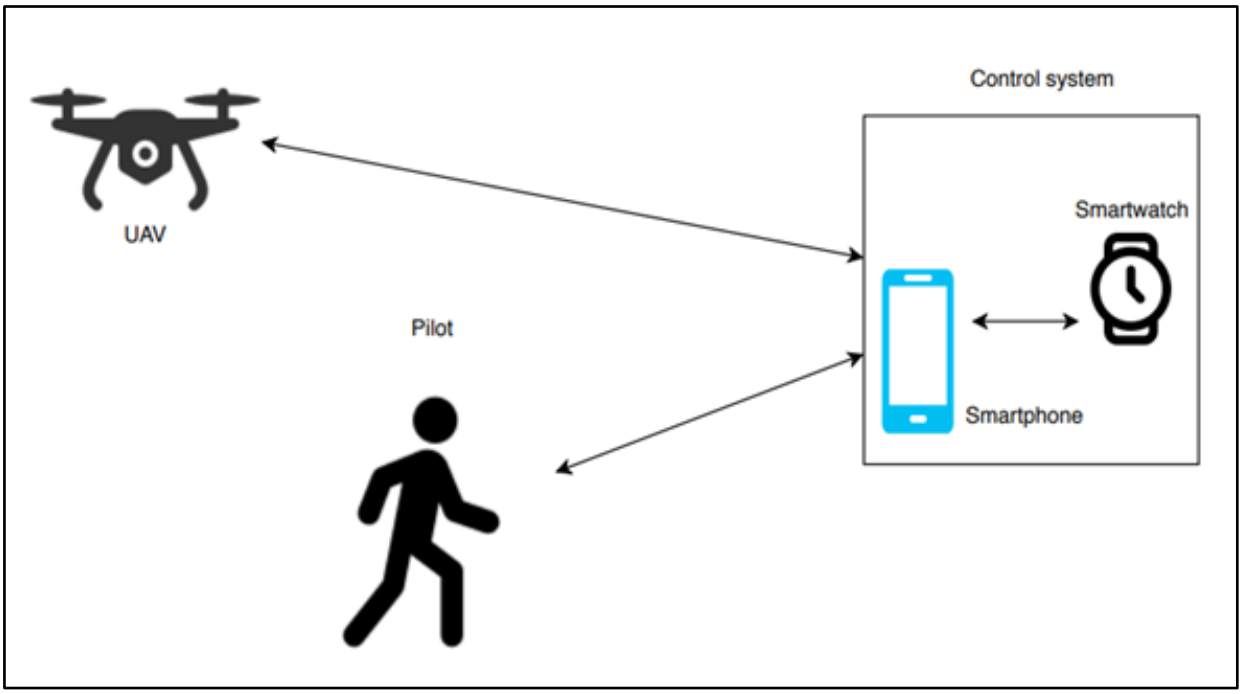


Figure 4-5: High-level visual representation of the UAS control architecture with a smartwatch

Figure 4-6 is a visual representation of how the pilot needs to move his/her arm to control the pitch, yaw and roll of the UAS (Fernandez *et al.*, 2016). In the figure, the positive and negative values for each instruction are indicated by the plus and minus signs. This means, for example, that when the pilot moves his/her arm upwards, the value of pitch will increase, and if the pilot moves his/her arm downwards, the pitch will decrease. The UAS will move upwards when the

pitch is increased and downwards when the pitch is decreased. This enables the pilot to control the UAS without the need of having the smartphone in his/her hand.

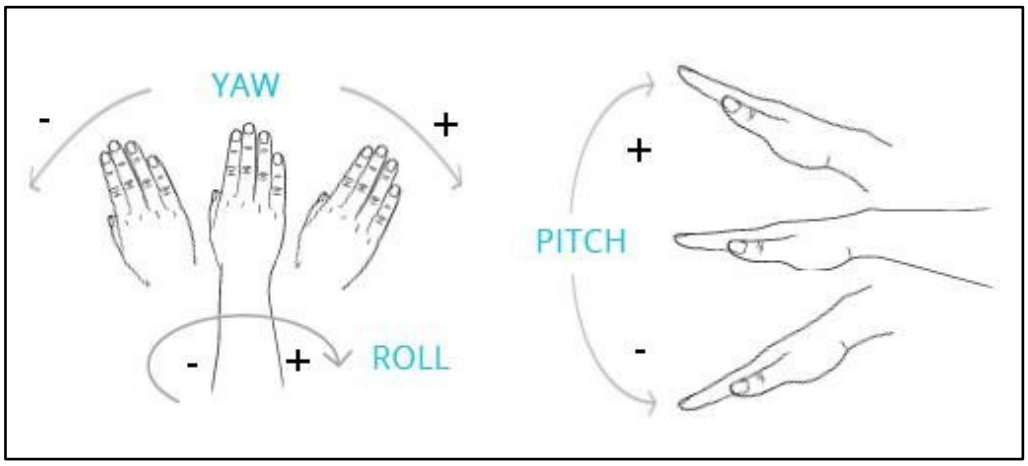


Figure 4-6: Visual representation of yaw, pitch and roll arm movement

Table 4-1 denotes sample accelerometer data that was gathered for yaw movement. In the first row is the type of movement, in the second row, are the indicators for the positive (increase) and negative (decrease) movements and in the third row, are the axes indicators. The rows that follow are the actual accelerometer data for each movement. The negative values were gathered when the pilot moved his/her arm to the left, and the positive values were gathered when the pilot moved his/her arm to the right. The negative values do not necessarily mean that the accelerometer data will be negative; it means that there is a decrease in the accelerometer value from the original position. For positive values, the values are increasing, but will not necessarily be positive.

Yaw					
-			+		
x	y	z	x	y	z
-2.4645	0.00958	9.10754	2.81403	-3.8004	9.54329
-2.4836	0.56982	8.82102	2.45969	-3.2992	9.8258
-2.7853	0.36948	8.81545	2.14844	-4.0366	9.48182
-2.905	0.11647	8.98706	2.23942	-3.5187	9.41323
-2.0638	0.21548	9.20131	2.48919	-3.0877	9.48471
-2.4181	0.44053	9.0748	2.32238	-3.6344	9.23204
-2.8748	0.04866	9.02368	2.63285	-3.5961	9.80587
-2.8748	0.83797	9.56946	2.12049	-3.6152	9.95287
-2.8199	0.06859	9.85453	2.46046	-2.9497	9.97848
-2.8112	0.16915	9.38048	2.87628	-3.3695	10.0804
-2.9337	0.1883	9.92714	2.88586	-3.7047	10.3493

Table 4-1: Yaw sample accelerometer data obtained from the smartwatch

The sample accelerometer data for the roll movement is presented in Table 4-2. Negative roll movement is done by tilting the arm to the left, and positive roll is achieved by tilting the arm to the right.

Roll					
-			+		
x	y	z	x	y	z
-8.805874	1.139640	-4.678270	-9.102755	1.556231	-2.662352
-8.537724	1.115698	-4.663905	-9.515114	1.465251	-2.762909
-8.408437	0.828394	-4.735731	-9.572019	1.259350	-2.796428
-8.757990	0.694319	-4.845864	-9.342176	1.422156	-2.614468
-8.781932	0.976834	-4.860230	-9.279925	1.422156	-2.595315
-8.513782	0.191536	-4.912902	-9.227253	1.374272	-2.628834
-9.073688	0.300763	-4.726931	-9.206987	1.283292	-2.695871
-9.028483	0.750873	-4.525041	-9.461885	1.278504	-2.892196
-9.072472	0.974375	-4.685065	-9.212888	1.345541	-2.777274
-8.556100	1.039084	-4.620033	-8.963891	1.412579	-2.662352
-8.370130	1.245761	-4.673482	-8.714894	1.479617	-2.547431

Table 4-2: Roll sample accelerometer data obtained from the smartwatch

In Table 4-3 is the sample accelerometer data for the pitch movement. A positive pitch movement is made when the pilot moves his/her arm upwards, and a negative pitch movement is made by moving the arm downwards.

Pitch					
-			+		
x	y	z	x	y	z
-2.331953	-6.426037	7.163452	-1.143652	-1.215478	9.447520
-2.370260	-6.239290	7.321469	-1.216255	-1.177947	9.394848
-2.422932	-6.694188	7.125145	-1.503559	-1.120486	9.083601
-2.001553	-6.785168	7.096414	-0.900220	-1.043872	9.155427
-2.451663	-5.679047	7.053318	-0.928950	-1.909797	9.308656
-2.432509	-6.493075	7.268796	-1.077391	-1.465251	9.744401
-2.470816	-5.640739	7.905654	-1.000776	-1.474828	9.653421
-2.159570	-6.761226	7.125145	-1.096544	-1.335965	9.390059
-2.442086	-6.282385	7.503428	-1.173159	-1.781286	9.409213
-2.384625	-6.617573	7.158663	-1.240197	-2.001553	9.481039
-2.480393	-6.631939	7.201759	-1.297657	-1.829170	9.744401

Table 4-3: Pitch sample accelerometer data obtained from the smartwatch

One of the goals of this study is to integrate simplified user control into a UAS control system. Controlling the UAS with a smartwatch, instead of the smartphone, simplifies UAS control. The pilot does not need to have a smartphone in his/her hand to control the UAS. Although it is

possible to control the UAS with the smartwatch by using the accelerometer data, the problem is that it is difficult to control the UAS while the pilot is moving. The UAS moves as the pilot moves his/her arm because the accelerometer data gets updated when the smartwatch is moved. In the next section is a discussion on simplifying UAS control to possibly solve this problem.

4.3 Improved UAS control with gesture recognition implementation

Various UAS control techniques are investigated in Chapter 3 (Section 3.2.2), and one of the control techniques that is investigated in the study is gesture control. With gesture control, as mentioned in Section 3.2.2.4, the pilot can wear a device on the arm or wrist, and accelerometer data can then be used to capture arm, hand, and also possible finger gestures. This was also demonstrated in the previous section. The system implemented in this study uses accelerometer data with a radial basis function neural network (RBFNN) to control the UAS. Gestures are used instead of using accelerometer data directly to control the UAS. The system uses a smartwatch to capture gesture data which is then sent to the smartphone. With the use of an RBFNN that performs inference on the phone, the gesture is recognised. Predefined rules are then used to map these gestures to various control tasks that can be performed by the UAS. These gestures are used to control the UAS instead of controlling the UAS with the remote control. Figure 4-7 is a high-level visual representation of the UAS control architecture that uses integrated gesture control to simplify and automate UAS control.

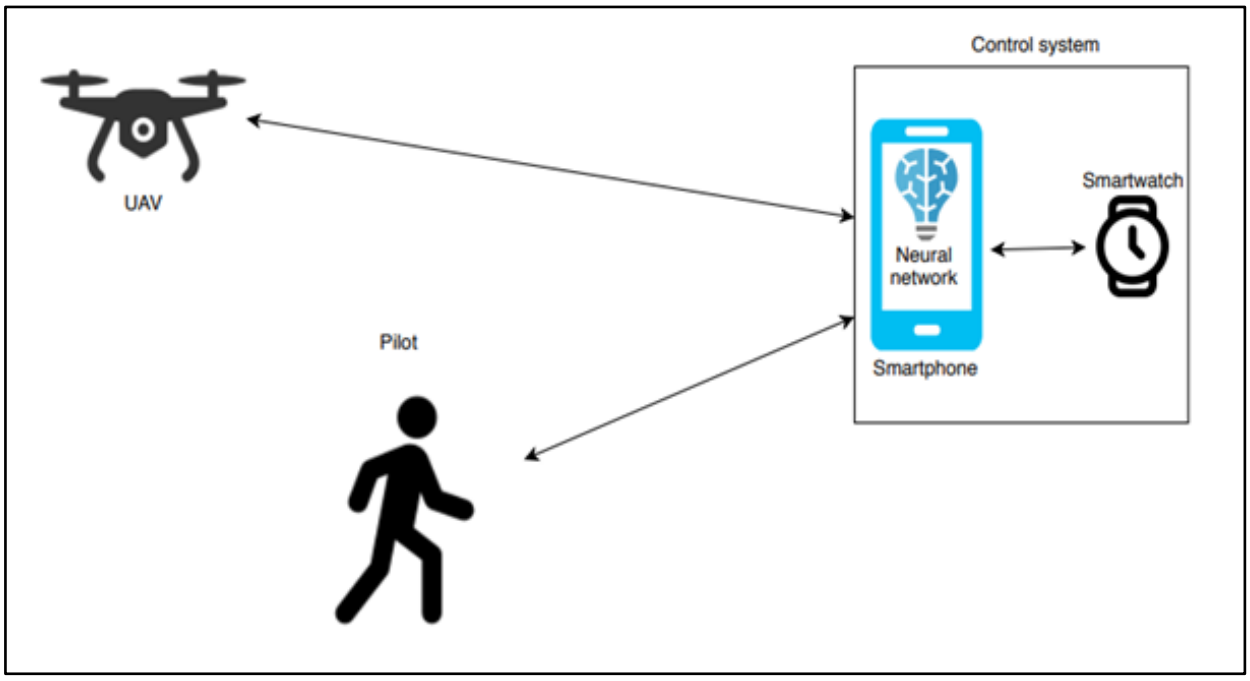


Figure 4-7: High-level visual representation of UAS control architecture with integrated gesture control

Improved UAS control with the gesture recognition system is implemented in five steps. The first step is to identify the gestures that will be used for UAS control. After the gestures are identified, the accelerometer data for these gestures are captured. The third step is to apply a Fast Fourier Transform (FFT) analysis on the gesture accelerometer data. In Section 3.3.7, it is mentioned that an FFT can be used to analyse periodic data. The accelerometer data is transformed into the frequency domain, which then shows how the frequencies change over time. After the accelerometer data is processed, the fourth step is implemented by using the data to train the RBFNN model. This model is then subsequently used for gesture recognition. The last step is to integrate the gesture recognition system to simplify UAS control. The next section is a discussion on the control gestures that are used for this system.

4.3.1 Control gestures

In this section, gestures are identified that can be used to control the UAS. Instead of controlling the UAS directly with the accelerometer data, the data are mapped to the gestures by the RBFNN. In this study, different gestures were investigated and evaluated that could be used for gesture recognition. Based on the evaluations, the following three gestures were identified and used for gesture recognition: up and down, circle, and up then right and left gesture. These gestures are visually presented in Figures 4-8, 4-9, and 4-10. In Figure 4-8 is the first (up and down) gesture. To perform this gesture, the pilot moves his/her arm in an upward and then downward motion.

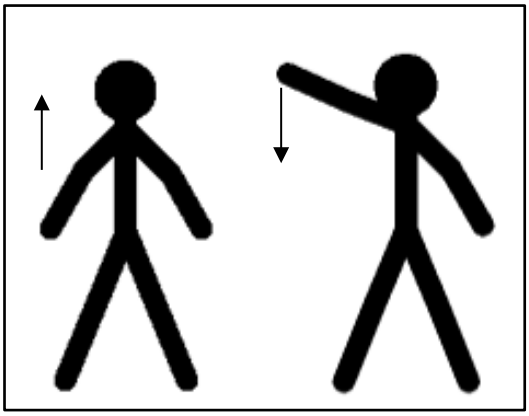


Figure 4-8: Gesture 1 - Moving arm up and down motion gesture

Figure 4-9 denotes the second (circle) gesture. The pilot moves his/her arm in a circular motion, starting from the left-hand side in a clockwise direction.

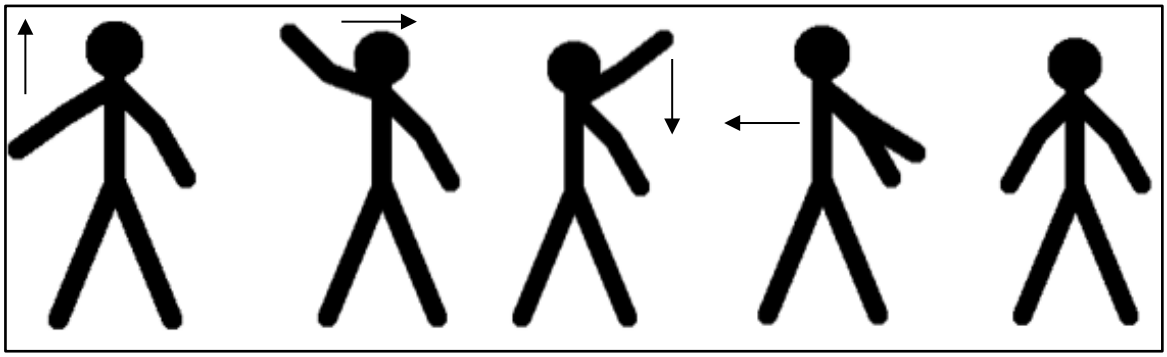


Figure 4-9: Gesture 2 - Moving arm in circular motion gesture

The third (up and then right and left) gesture is shown in Figure 4-10. To perform this gesture, the pilot moves his/her hand in an upward motion, to the right and lastly to the left. This gesture forms a T-shaped arm motion.

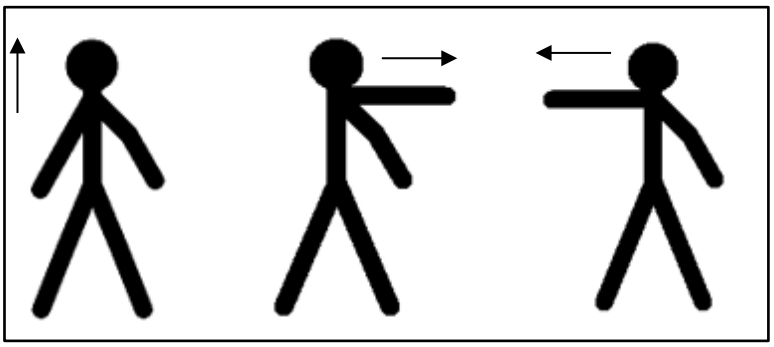


Figure 4-10: Gesture 3 - Moving arm up and then right and left motion gesture.

For the experiment to work, the smartwatch should be worn on the arm that is used to make the gesture. After the gestures are identified, the example accelerometer data must be captured for each gesture. This data acquisition is discussed in the next section.

4.3.2 Acquiring example accelerometer data for gesture recognition

The accelerometer data is acquired from the Moto 360 smartwatch that is worn on the pilot's wrist. When the pilot presses the capture gesture button on the smartwatch, the pilot's arm movements are captured by recording the accelerometer data of the smartwatch for five seconds. Table 4-4 is a sample of the accelerometer data measured in metre per second squared (m/s^2). In the first column is the gesture that was performed. In this case, it was the first gesture which is the up and down arm motion. In the second column is the iteration count of gesture one. It indicates that this is the first reading of gesture one. In the last three columns are the actual accelerometer data and each column indicates from which axis the accelerometer data is captured. Multiple iterations of each gesture are recorded to provide sufficient training data for the RBFNN.

Gesture	Iteration	x-axis	y-axis	z-axis
1	1	-1.67115	-0.70390	-9.75398
1	1	-1.73819	-0.78051	-9.77792
1	1	-1.73819	-0.66080	-9.66779
1	1	-1.76692	-0.73741	-9.69173
1	1	-1.67594	-0.64643	-9.58638
1	1	-1.58496	-0.82361	-9.72046
1	1	-1.77650	-0.78530	-9.73482
1	1	-1.70946	-0.89064	-9.69652
1	1	-1.75734	-0.74220	-9.66779
1	1	-1.60412	-0.89543	-9.73482
1	1	-1.72383	-0.87628	-9.63427
1	1	-1.87227	-0.86191	-9.69173
1	1	-1.86748	-0.92416	-9.68215

Table 4-4: Sample of accelerometer data that is obtained from the smartwatch

Table 4-5 is a summary of the complete data set. The total number of data points per axis for gesture one was 16700, which gives an average of 167/5 readings per second (Hertz). This means that the measurements were taken at intervals of approximately 5/167 seconds each. Similar calculations can be made for the second and third gestures.

Gesture	Iterations	Axis	Iterations per axis	Data points per axis
1	100	x	100	16700
		y	100	16700
		z	100	16700
2	100	x	100	13900
		y	100	13900
		z	100	13900
3	100	x	100	15400
		y	100	15400
		z	100	15400
Total	300		900	138000

Table 4-5: Summary of complete data set of the accelerometer data for gesture control

The next step is to apply an FFT analysis to the accelerometer data of Table 4-5. The FFT adapts the accelerometer data so that it can be used by the RBFNN and is discussed in the next section.

4.3.3 Accelerometer data processing

MATLAB™ was used for converting the accelerometer data into FFT values, and the visual result is shown in Chapter 5. A sample of the accelerometer data that was converted into FFT values is presented in Table 4-6.

Gesture	Iteration	x-1	x-2	x-3	y-1	y-2	y-3	z-1	z-2	z-3
1	1	472.89	790.59	225.79	561.19	11.62	211.55	715.35	238.31	245.59
2	1	481.83	449.07	197.96	353.97	65.08	88.19	467.87	199.53	106.61
3	1	307.42	68.83	44.79	127.69	35.73	32.52	232.48	71.11	45.01
1	2	506.59	623.36	236.15	461.44	59.92	139.84	601.36	182.96	188.44
2	2	420.96	410.94	189.35	294.16	65.82	58.39	359.73	161.84	87.76
3	2	446.21	111.43	41.67	168.95	82.83	112.82	392.30	100.20	14.05
1	3	644.54	666.96	284.99	423.35	59.95	135.99	614.46	271.95	179.07
2	3	433.75	430.42	191.93	360.91	46.75	79.38	405.97	170.11	132.71
3	3	244.40	115.09	88.48	109.51	149.34	84.35	393.62	88.07	50.57
1	4	438.94	688.45	203.20	420.11	27.42	141.33	681.52	169.89	259.56
2	4	437.08	393.48	182.73	252.47	38.85	98.09	384.03	169.26	98.16
3	4	239.77	125.32	61.62	130.14	94.27	58.90	376.42	66.17	46.88
1	5	466.60	665.68	198.87	538.22	6.07	129.17	708.46	175.88	226.30
2	5	483.71	413.22	190.50	351.31	37.74	115.84	385.79	188.75	85.12
3	5	189.81	84.91	63.64	93.93	107.25	69.09	319.05	66.74	34.37

Table 4-6: Sample of accelerometer data that were converted into FFT values

The first column indicates the gesture that is performed, in the second column is the iteration of the gesture and in the columns that follow, are the first three FFT values for the *x*, *y*, and *z*-axis of the accelerometer measurements respectively. After the data is processed, it can be used to train the RBFNN model, which is discussed in the next section.

4.3.4 Training the RBFNN model

After the gestures have been identified, and the accelerometer data has been converted into FFT values, the next step is to use the data to train the RBFNN model. A visual representation of the RBFNN architecture that is used for this experiment is shown in Figure 4-11. This RBFNN consists of the input layer, hidden layer, and output layer. The input layer comprises the accelerometer data of each gesture that was converted into FFT values. This means that each input will consist of the nine FFT values, as shown in Table 4-6 in the previous section.

A process of trial-and-error was followed to determine the best number of hidden nodes. Eight hidden nodes were eventually chosen for training the model. The output layer should indicate the gesture that is associated with the FFT values of the input layer. The output layer has three

nodes, and each node should return a probability. These output probabilities are associated with a specific gesture identified in Section 4.3.1. Each gesture was captured for a total of one hundred times (iterations), and the first three FFT values of each axis (*x*, *y*, and *z*) were used per gesture. In total, it gives nine FFT values per iteration. This means that the data set consisted of one hundred data points per gesture and three hundred data points in total. Seventy per cent of the data was used to train the RBFNN model, and twenty per cent of the data was used for the validation set. The last ten per cent of the data was used to test the trained RBFNN model. The results obtained are discussed in Chapter 5.

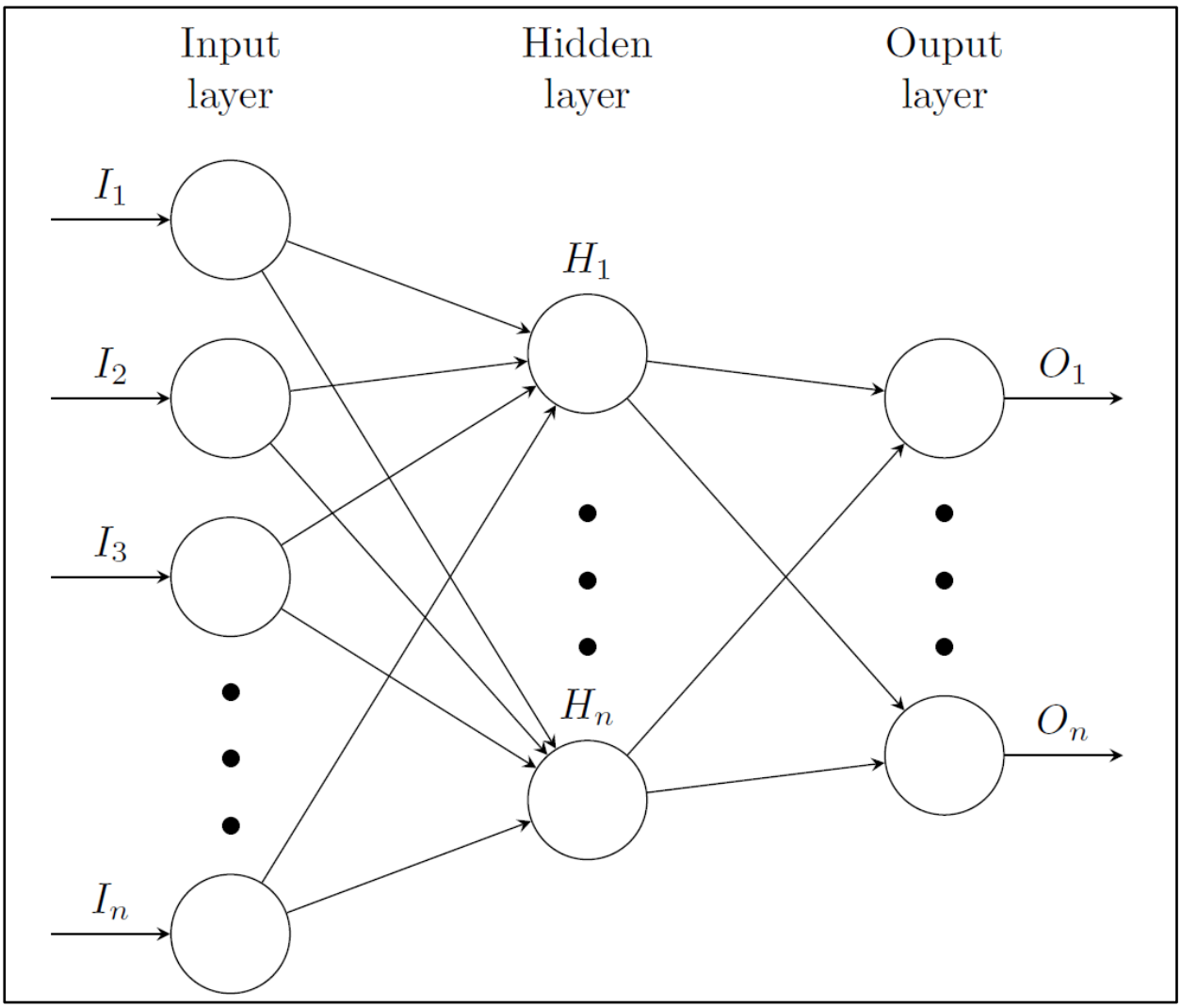


Figure 4-11: Visual representation of the RBFNN architecture

The computational power of the hand-held and wearable devices is limited when compared to traditional computers (desktop and laptop). It can cause a delay in performance when these devices are used for training the RBFNN model. The RBFNN model is therefore trained using a traditional computer and then exported to speed up the process. This process enables the trained model to be loaded on the hand-held device instead of training the model on the device itself. The computer that was used to train the RBFNN model is an Intel 4200H 64bit desktop

computer, running the Windows 10 operating system, with 8GB RAM, an NVIDIA GTX 950 M graphics card, and a Samsung 750 EVO 250GB SSD. Training the model on a traditional computer also provides the advantage of visually inspecting the results which are discussed in Chapter 5. After the RFBNN model is trained and integrated into the sample application, it can be used for gesture recognition. In the next section is a discussion on how gesture recognition is integrated to simplify and improve UAS control.

4.3.5 Integrating gesture recognition

Figure 4-12 is a process flow diagram of the integrated gesture control implementation. On the left side of the diagram is the component that is responsible for these specific tasks and on the right side is the processes flow of the task performed. The gesture is captured when the pilot clicks on the gesture button on the smartwatch which is connected to a hand-held device. The data is gathered from the accelerometer for five seconds and is then processed by applying an FFT analysis on the data. After the data is processed, it is sent to the RBFNN that is executing on the smartphone for gesture recognition. The trained RBFNN model that was loaded on the smartphone is then used to determine the type of gesture that was performed by the pilot. The gesture result can then be used to send control instructions to the UAS, based on the predefined rules that are built into the application. For example, if the pilot moves his/her wrist up and down the UAS will move upwards.

This experiment is used to simplify UAS control by enabling the pilot to use his/her arm to control the UAS instead of having the smartphone in his/her hand. When the pilot does not need to have his/her smartphone in his/her hand, it enables the pilot to do other things. Unlike the experiment where the accelerometer data is directly used for controlling the UAS, the UAS does not move when the pilot moves his/her arm because the accelerometer data is not used to control the UAS. Instead, it is used to record the gesture which is mapped to predefined control instructions. The rules can also be associated with more sophisticated control instructions. In the next section is a discussion on an example of a more complex implementation that enables the UAS to follow the pilot autonomously with the help of GPS information.

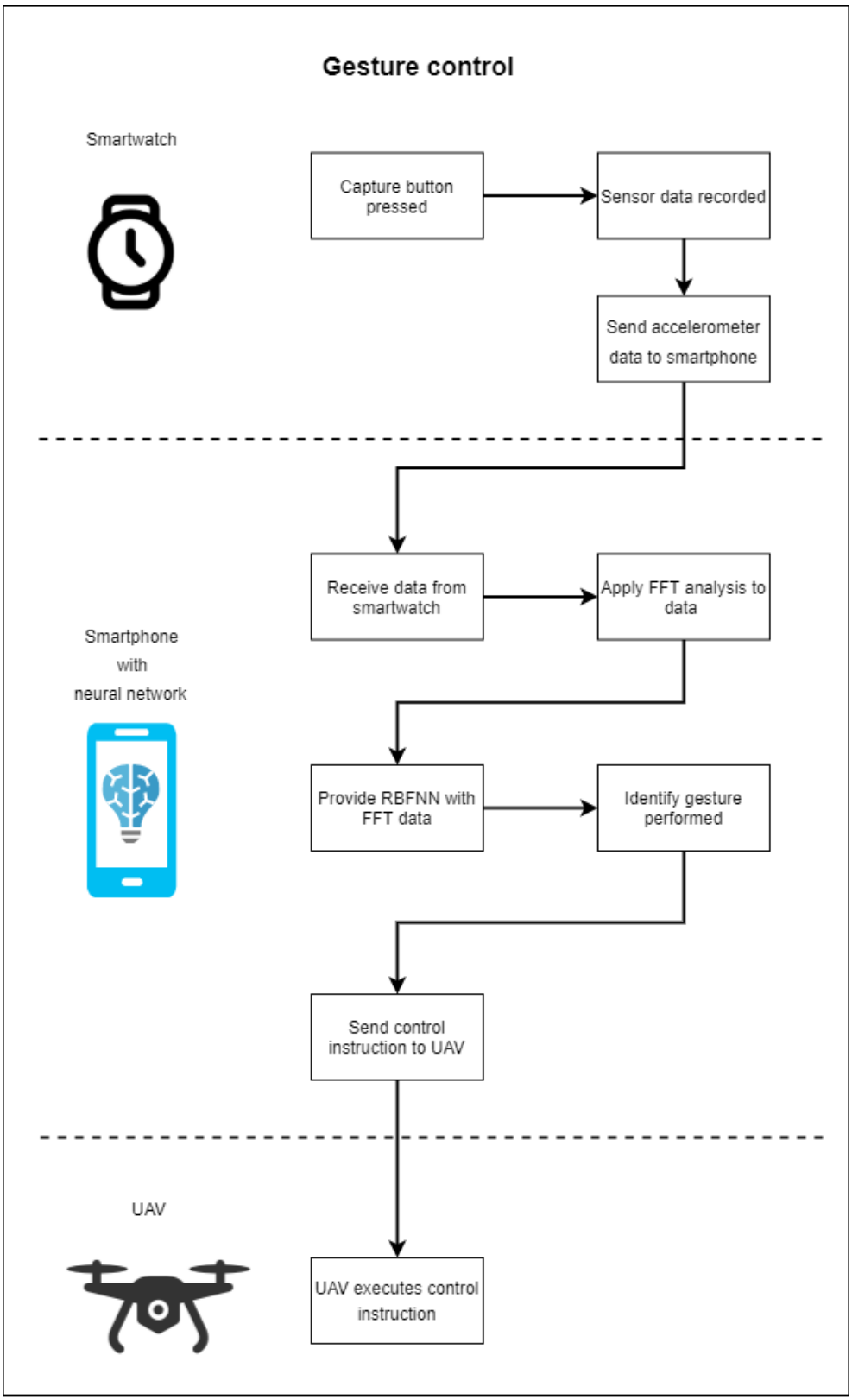


Figure 4-12: Process flow diagram of integrated gesture recognition

4.4 UAS follow function

Enabling the UAS to follow the pilot autonomously can simplify UAS control. When the UAS follows the pilot autonomously, the pilot can focus on other activities instead of controlling the

UAS manually to follow him/her. In Section 3.2.3.7, it is mentioned that a GPS can be used for object-location tracking. A simple rule-based method is implemented to enable the UAS to autonomously follow the pilot by using GPS coordinates; this is visually shown in Figure 4-13. In this experiment, the GPS coordinates of the pilot's hand-held device are sent to the UAS. The method uses the GPS coordinates and calculates the distance between the UAS and the pilot through the haversine formula (Bujari *et al.*, 2012). This distance calculation helps to prevent the UAS from crashing into the pilot if the pilot is moving slowly or standing still. This calculation works as follows: if a distance of ten metres is used, it will follow the person from ten metres away. If the UAS comes within or less than ten metres of the pilot, it will go into a hovering state and will follow the pilot again, once the distance between the pilot and the UAS is more than ten metres. The following function can be initiated by mapping a gesture to the function. The UAS uses piloting states to determine the current piloting behaviour of the UAS. These piloting states help to prevent the UAS from performing an "illegal" action. For example, the UAS cannot change to the hovering state if the UAS is currently on the ground. The UAS first needs to be in a taking-off state before it can transition into a hover state.

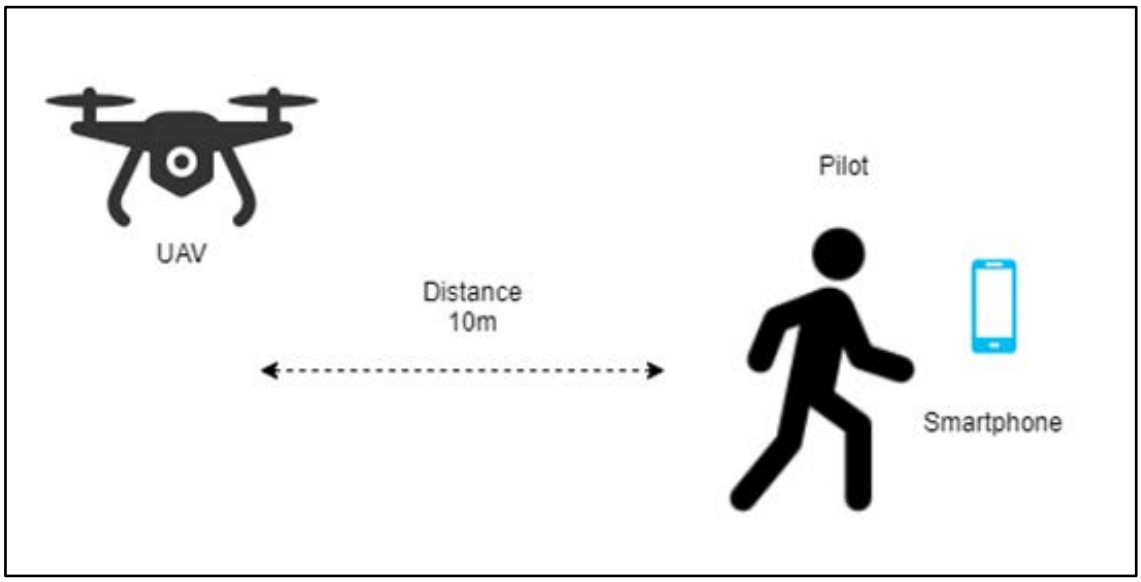


Figure 4-13: High-level visual representation of UAS follow function

In Figure 4-14 is a process flow diagram of the piloting states used by the UAS for this implementation. The following function was added as a state to ensure the UAS can safely transition into the following state. It is also worth noting that the UAS can only move into the flying state after the UAS has successfully entered a hover state. The solid lines indicate the flow between possible piloting states, and the dotted lines indicate the process flow towards the emergency state. When something goes wrong, the UAS moves into the emergency state.

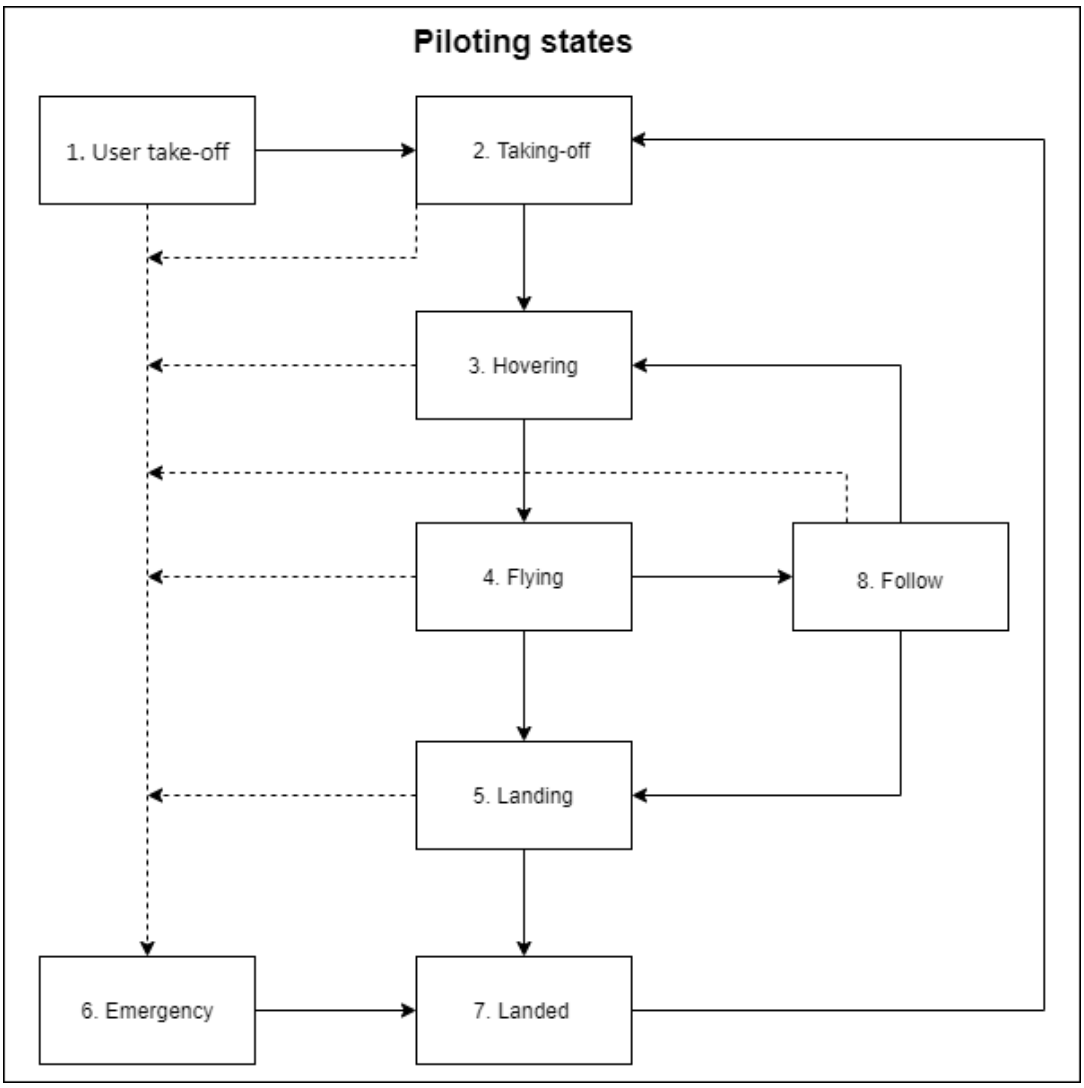


Figure 4-14: Piloting states process flow diagram

Table 4-7 is an overview of the piloting states and a description of the behaviour of each piloting state.

Nr.	State	Behaviour
1	User take-off	Waiting for the pilot to initiate take-off action
2	Taking-off	Occurs after the user has initiated take-off and the UAS is lifting off the ground
3	Hovering	The UAS is hovering and not moving in any specific direction
4	Flying	The UAS is moving either forwards, backwards, left, right, up or down
5	Landing	The UAS is preparing to land on the ground
6	Emergency	The autopilot takes over, and all other flying commands are ignored
7	Landed	The UAS has landed on the ground
8	Follow	The UAS follows the pilot, based on GPS coordinates

Table 4-7: UAS piloting states

The follow function simplifies UAS control by enabling the UAS to follow the pilot autonomously. There is no need for the pilot to use a smartphone to control the UAS to follow him/her continuously. UAS control is automated by using sensor data to navigate through the air instead of being piloted by remote control. Although the follow function does enable the UAS to follow the pilot autonomously, the speed of the UAS still needs to be manually adjusted if the pilot wants to change it. One possible method to automate the speed of the UAS is investigated and implemented in the next section.

4.5 Improved UAS control with automated activity recognition

In the gesture recognition implementation, gestures were used to automate various high-level UAS control tasks. In this experiment, a similar approach is used in an attempt to automate the speed of the UAS. The system is implemented by using a smartwatch to capture data associated with various activities (in terms of movement) performed by the pilot. The data is processed by an RBFNN to autonomously identify the activity that was performed by the pilot and to adjust the speed of the UAS accordingly. This process is done in four steps. The first step is to identify the activities and gather example data for the activities. In the second step, the data is pre-processed. In the third step, an RBFNN is used to train the activity recognition model. The last step is to integrate the activity recognition experiment with the UAS. In the next section, the data gathering process is discussed.

4.5.1 Acquiring data

Before the data is gathered, the activities that are going to be used for activity recognition are identified. Three activities are used for this experiment and are as follows: stand, walk and run. These activities are associated with what the pilot is currently doing in terms of movement. The data is gathered from the accelerometer (*x*, *y*, and *z*-axis values) of the smartwatch while the pilot is performing an activity. In the previous experiment, the data were collected for five seconds. Five seconds was the maximum time needed to capture any of the gestures. In this experiment, a buffer of sixty-four measurements is used to collect the data instead of a predetermined time limit. This buffer ensures that the same amount of data is collected for each activity and that the data collected is a power of two ($2^6 = 64$). The specific FFT algorithm that is used in this implementation requires a buffer that is a power of two. This technique is further discussed in Section 5.4.2. Table 4-8 is an example of the accelerometer data collected. Exactly sixty-four readings of each accelerometer axis (*x, y,* and *z*) were collected for each iteration. In the first column is the activity. In this case, it was the first activity (stand) that was performed. The second column is the iteration for this specific activity. For each activity, one hundred iterations were performed. In this example, it is the first iteration of activity one. The third column indicates the reading number for the iteration. It starts at one and goes to fifty. This means that

fifty readings of each accelerometer axis (x, y, z) were collected for each iteration. It gives a total of one hundred and fifty accelerometer readings per iteration. In the last three columns are the accelerometer data for each axis.

Activity	Iteration	Reading	x	y	z
1	1	1	-3.542840	-2.101360	8.846017
1	1	2	-3.533840	-2.113360	8.856217
1	1	3	-3.793970	-3.251330	8.536947
1	1	4	-3.164230	-0.359130	9.466674
1	1	5	-3.792400	-1.643870	8.799416
1	1	6	-4.782710	-1.000780	8.389840
1	1	7	-3.573600	-1.810010	8.936160
1	1	8	-3.847300	-1.944090	8.974141
1	1	9	-3.182500	-1.684510	9.156217
1	1	10	-3.134820	-1.304120	8.995430
1	1	.	.	.	.
1	1	64	-3.104900	-1.508900	9.199766

Table 4-8: Sample of accelerometer data of the activity

Table 4-9 is a summary of the complete data set for the accelerometer data.

Activity	Iterations	Axis	Iterations per axis	Data points per axis
1	100	x	100	6400
		y	100	6400
		z	100	6400
2	100	x	100	6400
		y	100	6400
		z	100	6400
3	100	x	100	6400
		y	100	6400
		z	100	6400
Total	300		900	57600

Table 4-9: Summary of complete data set of the accelerometer data for activity recognition

The next step is to process the data that was collected and this is discussed in the next section.

4.5.2 Data processing

Data gathered in the previous step is obtained in three different axes (*x*, *y*, and *z*). In this step, the data from the three different axes are transformed into a single scalar value. The motivation

for this is that the smartwatch could be in different positions on the pilot's body, which may influence the results (Bujari *et al.*, 2012). The reason for not applying this method to the gesture recognition experiment was that the smartwatch needs to be in a specific position to be able to perform the gesture. If the smartwatch was not in the same relative position, the wrong gesture might be identified. The three-axis values are transformed into a single scalar value by calculating the magnitude of the accelerometer vector with the use of the equation

$$v = \sqrt{x^2 + y^2 + z^2} \tag{4-1}$$

where *v* denotes the magnitude of the acceleration and *x*, *y* and *z* denote the acceleration values of the three axes respectively. By using Equation 4-1, the data becomes independent of the position of the smartwatch while the data was collected. An example of calculating the magnitude by using the accelerometer vector is as follows:

$$v = \sqrt{-3.54284^2 + (-2.10136)^2 + 8.846017^2}$$
$$= 9.758045\ m/s^2.$$

In Table 4-10, a sample of the data is shown after the magnitudes are calculated. The magnitude values in the third column of Table 4-10 were derived by applying Equation 4-1 to the sample data in Table 4-8.

Iteration	Reading	Standing	Walking	Running
1	1	9.75804519	10.21534900	9.97039530
1	2	9.76662148	10.22367700	10.04308400
1	3	9.89165402	10.23187200	10.12737800
1	4	9.98795487	10.27524900	10.03193600
1	5	9.72184799	10.16278700	10.08087300
1	6	9.70903140	10.27801000	9.99508740
1	7	9.79294178	10.24629400	10.07627700
1	8	9.95572247	10.14315600	10.10631100
1	9	9.83881039	10.25824500	10.01536800
1	10	9.61486278	10.17152100	10.06846400
1	.	.	.	.
1	64	9.82613239	10.27755500	10.09586800

Table 4-10: Sample of accelerometer data after magnitudes are calculated

After the magnitude is calculated for each accelerometer vector, an FFT analysis is applied to the data. In Table 4-11, a sample of FFT values that are calculated is shown. The first three FFT values for each iteration per activity are used to train the RBFNN model

Activity	Iteration	FFT value 1	2	3
1	1	5.078410	6.559039	4.004165
1	2	7.681911	6.176175	2.360158
1	3	1.574130	0.823663	2.228085
1	4	4.501405	6.979974	4.826355
1	5	5.381146	3.011971	2.605436
2	1	15.327155	32.879104	8.724568
2	2	3.728278	10.457231	11.915903
2	3	20.107205	11.447385	3.353977
2	4	50.420773	26.833764	25.024884
2	5	22.828280	28.263717	29.515372
3	1	27.963420	13.021914	12.951687
3	2	14.502067	19.082311	9.764349
3	3	46.770897	53.609531	28.651377
3	4	79.968221	45.254785	31.638573
3	5	55.047675	7.561278	23.675691

Table 4-11: Sample of FFT values after magnitude is calculated, and an FFT is applied

After the data has been processed, it can now be used in the training of the RBFNN model, which is discussed in the next section.

4.5.3 Training the RBFNN model

The training step of the activity recognition system is done similarly to the gesture recognition system. The input layer of the RBFNN comprises the first three FFT values of each activity. This means that each input will consist of three FFT values. As with the gesture recognition system, the best number of hidden nodes was found by the process of trial-and-error. Five hidden nodes were eventually chosen when training the model. The output layer should indicate the activity that is associated with the FFT values of the input layer. The output layer has three nodes, and each node should return a probability associated with a specific activity. Seventy per cent of the data was used to train the RBFNN model, and twenty per cent of the data was used for the validation set. The last ten per cent of the data was used to test the trained RBFNN model. Each activity was captured for a total of one hundred times. For each activity, the first three FFT values were used. This means that each activity consisted of one hundred data points, and the complete data set consisted of three hundred data points. Training for activity recognition was done with the same computer that was described in Section 4.3.4. After the RBFNN model for activity recognition was trained, it could be used for this task. In the next section is a discussion on integrating the activity recognition model with the UAS to simplify and improve UAS control.

4.5.4 System integration

Figure 4-15 is a process flow diagram of the integrated activity control implementation.

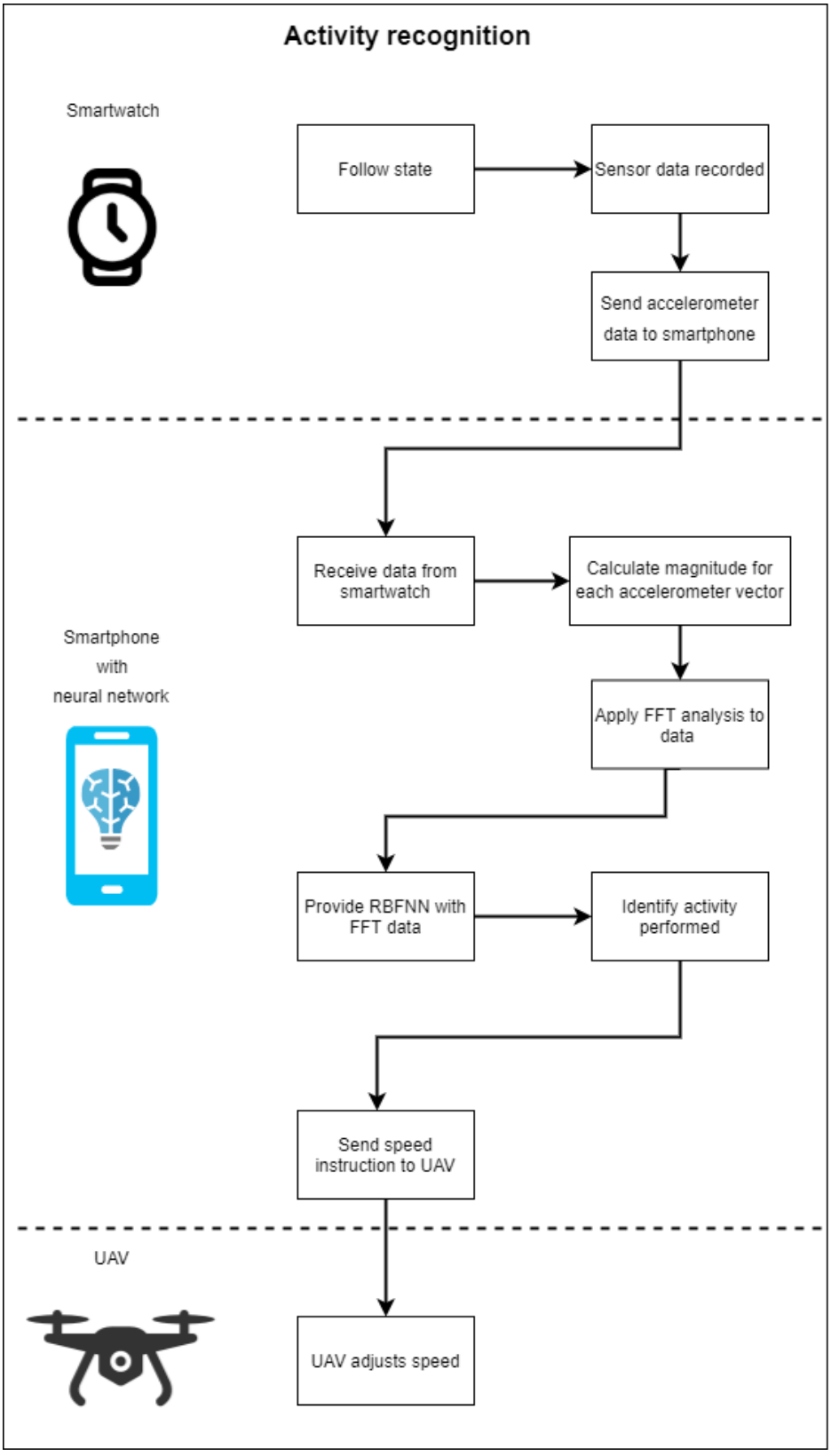


Figure 4-15: Process flow diagram of integrated activity control

While the UAS is in the following piloting state, the smartwatch will capture the accelerometer data autonomously. The data is captured every ten seconds after the UAS's speed has been adjusted. Capturing and adjusting the speed of the UAS only every ten seconds, reduces battery consumption of the smartwatch and smartphone. Sending the data continuously could negatively impact the battery life of these devices. It is possible to change the intervals for capturing the data for a specific-use case. After the data has been captured, the magnitude is calculated for each accelerometer vector. An FFT analysis is then performed on the data and sent to the RBFNN model, which determines the type of activity the pilot is performing. The speed of the UAS will be adjusted accordingly at that moment, based on the activity that was identified.

The activity recognition system simplifies UAS control and improves UAS automation by enabling the UAS to autonomously adjusts the speed of the UAS. The pilot does not need to adjust the speed of the UAS by using the smartphone. In the next section is the conclusion of the chapter.

4.6 Conclusion

By using the literature studied in Chapter 3, various systems were implemented in this chapter to simplify UAS control and to improve automation with the help of gestures and activity recognition and an RBFNN. In the first implementation, the basic control techniques for the UAS were explored. With the second implementation, the smartwatch was used with accelerometer data to control the UAS. In the third implementation, specific control tasks could be automated with the help of gesture recognition and an RBFNN. A follow function was implemented to enable the UAS to follow the pilot autonomously. The last implementation used activity recognition to adjust the speed of the UAS autonomously.

The gesture recognition and activity recognition systems were similar. However, the main difference between these two systems was that the activity recognition system did not require the pilot to perform any control instruction to adjust the speed of the UAS. The speed was autonomously adjusted when the UAS was following the pilot. With the gesture recognition implementation, the pilot first needed to perform the gesture to send the UAS control instructions. Each implementation provided its use case which is discussed further in Chapter 6. Sample code of these implementations can be seen in Annexure A. In the next chapter is a discussion on the results that were obtained from these systems that were implemented.

CHAPTER 5 RESULTS AND DISCUSSION

In this chapter, the results obtained from the systems implemented in Chapter 4 are presented and discussed. The systems are compared to the four levels of automation that were discussed in Chapter 3 to determine if UAS control is simplified and automation improved. The levels of automation that were previously discussed in Section 3.3.4 are as follows:

- **Level 1:** All high-level piloting is controlled manually.
- **Level 2:** Some high-level piloting is still controlled manually unless the piloting is switched over to automatic.
- **Level 3:** The UAS control is by default automatic but can be switched to manual piloting.
- **Level 4:** A fully autonomous system

The goal is to get to a higher level of automation. Reaching a higher level of automation simplifies UAS control and improves automation by reducing the number of control tasks that need to be performed by the pilot. When the tasks are automated in this case, the pilot can focus on other tasks instead of UAS control tasks. The type of application for which the UAS is used determines the task that needs to be automated. This study only focuses on simplifying UAS control and improving automation in general and not on a specific application. The tasks that will be used to measure the level of automation are as follows: take-off and landing, follow the pilot and adjusting the speed of the UAS.

The basic control system findings are presented in Section 5.1. In Section 5.2, is a discussion on the results from using the smartwatch accelerometer data to control the UAS. The third system, which uses gesture control to improve UAS control results, is discussed in Section 5.3. Section 5.4 is a discussion on the results obtained from the activity recognition system. The chapter is concluded in Section 5.5.

5.1 Basic control system findings

The basic control implementation in Chapter 4, Section 4.1, uses a remote control to control the UAS. In this implementation, a smartphone with the app developed by the designer of the drone was used as a remote control. The application only has basic control functionality. Most piloting is controlled manually except for take-off, landing, and emergency controls. Take-off and landing are automated, but the pilot needs to use the remote control to initiate the take-off or landing instruction.

There is no autonomous follow functionality built into the basic control system. If the pilot wants the UAS to follow him/her, the pilot will need to control the UAS manually. The speed of the UAS is also manually updated by the pilot, using the remote control. Based on the levels of UAS

automation mentioned in the previous section, this system uses level 1 automation. The basic control system provides the base that is required to control the UAS, and it also offers the necessary tools that are required to add functionality that can improve and simplify UAS control. In the next section, the results from controlling the UAS with smartwatch accelerometer data are discussed.

5.2 Controlling the UAS with smartwatch accelerometer data results

One of the first steps in simplifying and automating UAS control was to investigate the possible methods that could be used. One method that was identified in this study was to use the smartwatch to control the UAS, as discussed in Chapter 4, Section 4.2. The Moto 360 (2nd generation) smartwatch was used to control the UAS. Accelerometer data were gathered to control the yaw, roll, and pitch. In Figures 5-1, 5-2, and 5-3 are visual representations for 11 readings of the yaw, roll, and pitch movements. In each of these figures, the negative movement is shown by the solid lines, and the dotted lines show the positive movement. The values of the *x*-axis are shown by the red lines, the *y*-axis values are in blue, and the *z*-axis values are in green. A visual representation from the accelerometer data for yaw is presented in Figure 5-1.

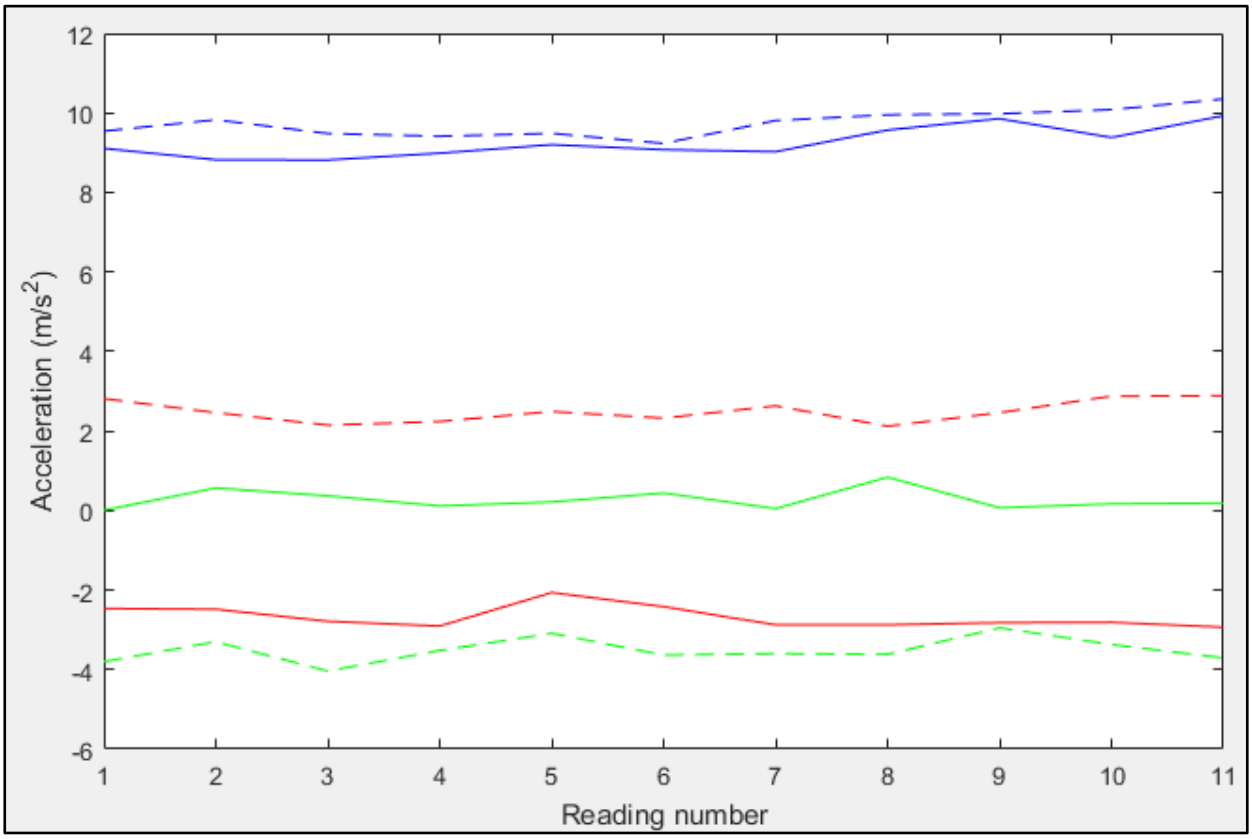


Figure 5-1: Accelerometer data comparison for yaw

From Figure 5-1, it can be seen that the *x*-axis shows the most significant change between the positive and negative values. This gives relatively the same effect as pushing the left stick on a

traditional UAS remote control (Section 3.1.3.1, Figure 3-3) to the left (negative) or the right (positive). The results from the accelerometer data for roll are shown in Figure 5-2. The biggest change can be seen on the *y*-axis. The roll movement is relatively the same as pushing the right stick on a traditional UAS controller to the left or right.

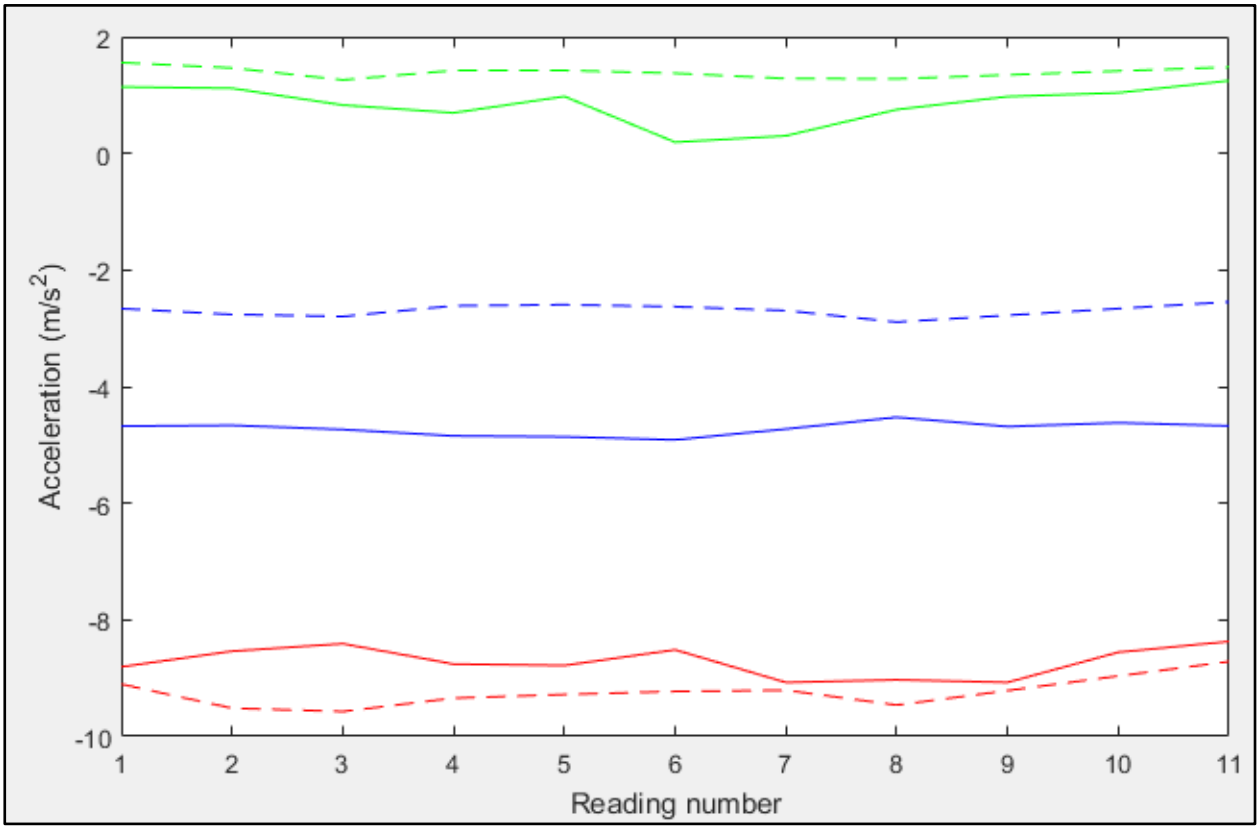


Figure 5-2: Accelerometer data comparison for roll

The results from the pitch movement can be seen in Figure 5-3. From Figure 5-3, it can be seen that the *z*-axis shows the most significant change between the positive and negative values. On a traditional UAS controller, this is relatively the same as pushing the right stick forwards or backwards. From Figures 5-1, 5-2, and 5-3, it can also be seen that there is sometimes a spike in the accelerometer data when compared to the other values. This is caused by the speed at which the arm is moved. The negative values also do not necessarily mean that the accelerometer data will be negative; it means that there is a decrease in the accelerometer value from the original position. For positive values, the values are increasing, but will not necessarily be positive. The advantage of this implementation was that the pilot could control the UAS without the need to have the smartphone in his/her hand. Control is simplified by using arm and wrist movements to control the UAS. This eliminates the need for controlling the UAS by using the smartphone directly or, in this case, trying to press the buttons on the smartwatch screen, which is relatively small. The disadvantage of this implementation is that it is difficult to control the UAS while moving, and the UAS moves as the pilot moves his/her arm because the

accelerometer data gets updated when the smartwatch is moved. To solve this problem, gesture recognition was investigated to control the UAS instead.

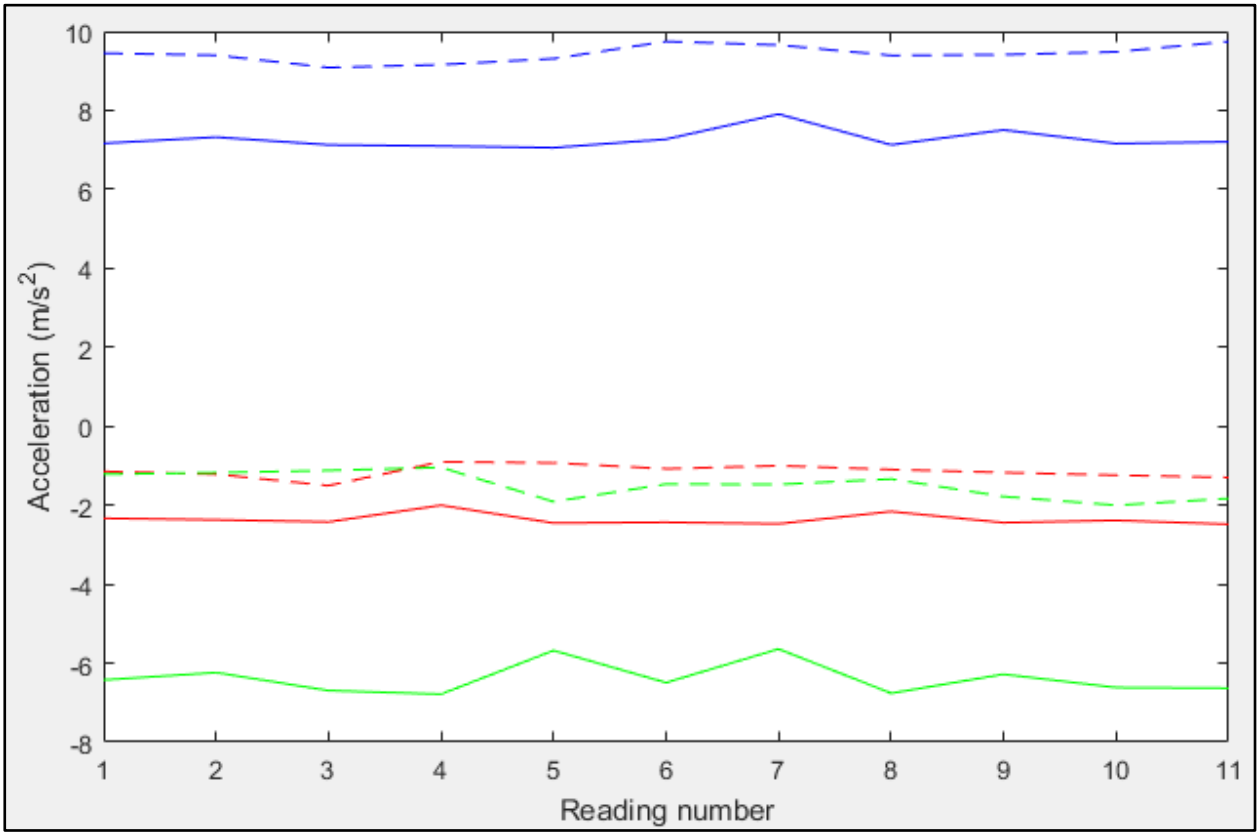


Figure 5-3: Accelerometer data comparison for pitch

The next section is a discussion on the results obtained from this implementation.

5.3 Improved UAS control system with gesture control results

The results from implementing gesture recognition with an RBFNN to control the UAS are presented and discussed in this section.

5.3.1 Control gesture results

In Chapter 4, Section 4.3.1, three gestures were identified to try and map accelerometer data to the gestures. The results from the gesture data should indicate that these gestures can be distinguished from each other. If the gestures are too closely matched, it can cause the wrong control task to be performed. Figure 5-4 is a visual representation for twenty samples of each axis of the accelerometer data gathered for gesture 1 (up and down). The acceleration of the *x*, *y,* and *z*-axis accelerometer data reading is shown on the *y*-axis. On the *x*-axis of Figure 5-4, the reading number is indicated. The *x*-value from the accelerometer reading is shown in red. Blue indicates the *y*-value, and green the *z*-value. From the *x*-value, it can be seen that the values increase and then decrease. This behaviour is similar to the previous experiment where the

accelerometer data was increased for positive movement and decreased for negative movement. The increase is caused by the upwards movement of the arm, followed by a decrease caused by the downwards movement of the arm.

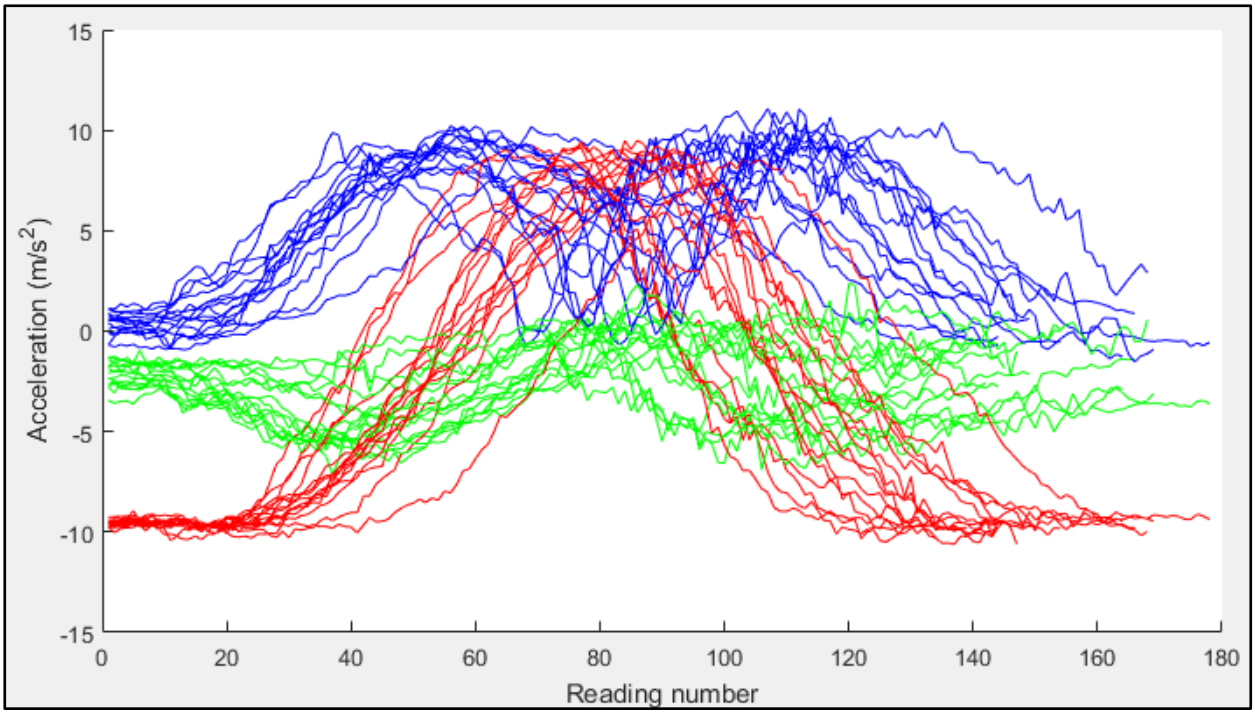


Figure 5-4: Accelerometer data of gesture 1 (up and down)

From the y-values, it can be seen that the values increase, decrease, and then increase and decrease again. This is caused by the arm extending forwards towards the middle of the movement and then back towards the top of the movement. When the arm is moving downwards, the same happens. The *z*-values should stay relatively unchanged, but from the results, it is clear that there is sometimes a small increase or decrease. This can be caused by the arm moving slightly in another direction while moving up and down. From Figure 5-4, it can be seen that it takes on average between 160 and 180 readings to perform gesture 1. The data lengths (number of readings) are essential for an FFT analysis and are later discussed in Section 5.3.2.

Twenty accelerometer data samples for each axis for the second gesture (circle) are shown in Figure 5-5. Between 120 and 140 readings are required to perform gesture 2. The accelerometer data for the *x*-value is shown in red, the *y*-value in blue, and the *z*-value in green. The *x*-value in Figure 5-5 is showing relatively the same pattern as the *x*-values in Figure 5-4. This is caused by similar upwards and downwards movements that are also made when performing the circle gesture. When performing the circle gesture, the pilot's arm also moves slightly to the right and left, which can be seen in the slight increase and decrease in the *y*-values. The *z*-value should stay relatively the same, but from Figure 5-5, it can be seen that

there are an increase and decreases in the *z*-values. This can be caused by the forward and backwards movement that might sometimes occur while performing the circle gesture.

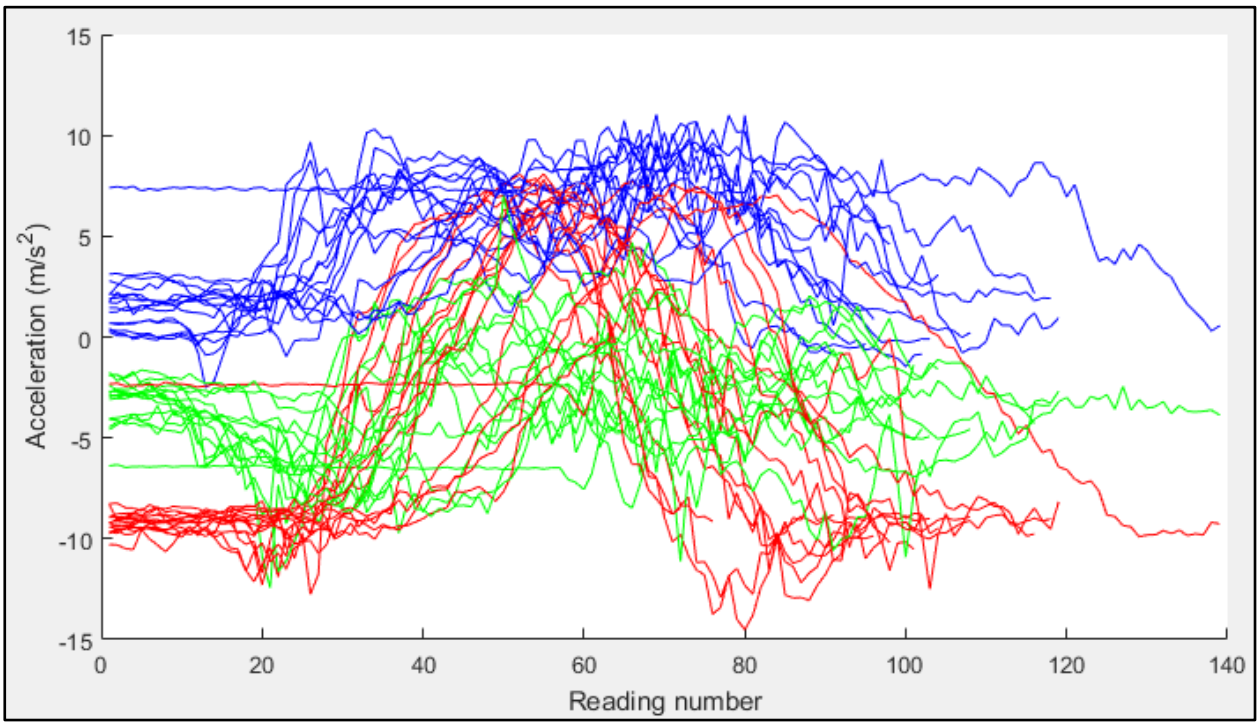


Figure 5-5: Accelerometer data of gesture 2 (circle)

Figure 5-6 is a visual representation of twenty sample accelerometer data readings for each axis from performing the third gesture (up and then right and left).

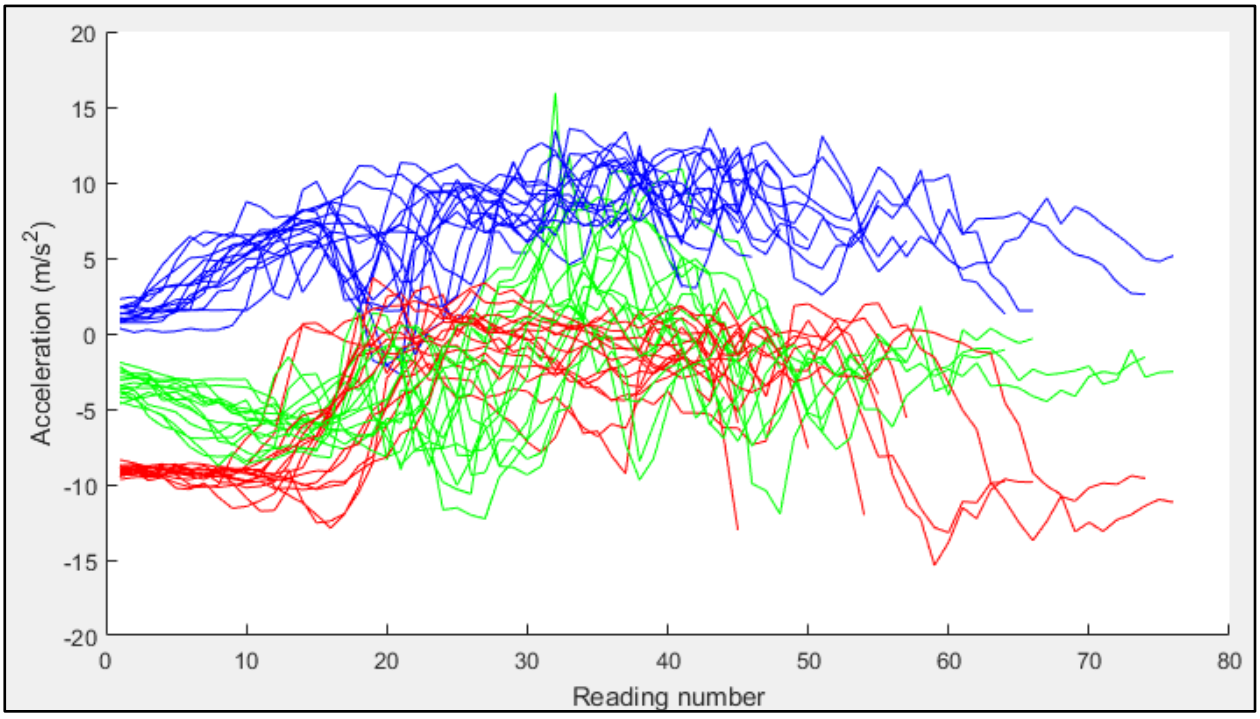


Figure 5-6: Accelerometer data of gesture 3 (up and then right and left)

It takes on average between 60 and 80 readings to perform gesture 3. The accelerometer data for the *x*-value is also shown in red, the *y*-value is in blue and the *z*-value is in green. From the *x*-value it can be seen that when the arm moves upwards, there is an increase, followed by a relative consistent acceleration when the arm moves right and then left and then a decrease when the arm moves downwards. The right and left arm movements can be seen in green. When the arm moves right, there is an increase in acceleration and when the arm moves to the left the acceleration decreases. The *y*-values should stay relatively consistent, but the arm could move forwards or backwards while performing the gesture.

Figure 5-7 is a visual representation of all three gestures in the same figure. The red lines represent gesture 1 (moving arm up and down motion), the green line shows gesture 2 (moving arm in a circular motion), and the blue line shows gesture 3 (moving arm up and then right and left motion). With the results obtained from the accelerometer data, the pilot should, for example, be able to perform the first gesture, and the result should be in the same range as the red lines. It is possible to see that visually each gesture is different from the other. It might not be as clear when using the accelerometer directly, and that is why an FFT analysis is applied to the accelerometer data. The FFT analysis transforms the data into the frequency domain, which then shows how the frequencies of the data change over time.

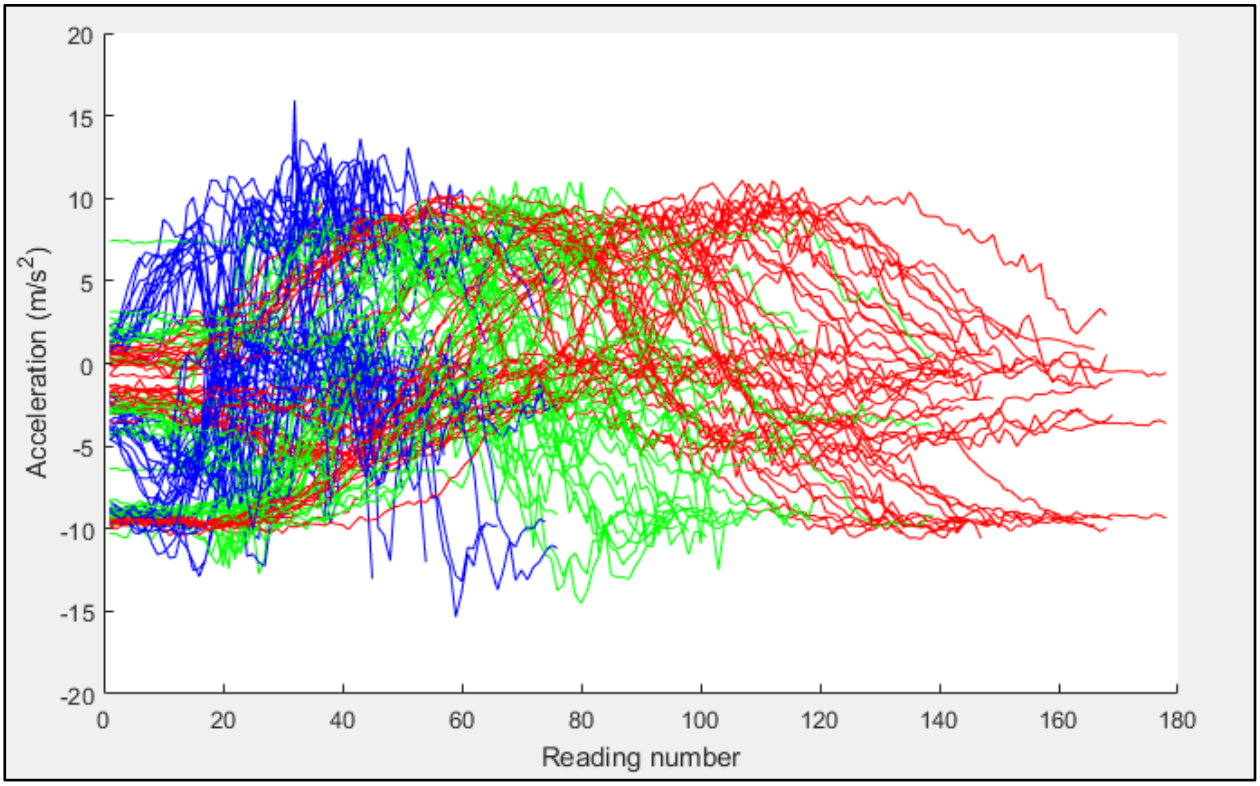


Figure 5-7: Results from accelerometer data for all three gestures

The result from the FFT analysis is presented and discussed in the next section.

5.3.2 Gesture recognition FFT accelerometer data results

Each gesture from which the accelerometer data was collected had different data lengths (number of readings). This is caused by the different length of time it takes to perform each gesture. For this specific experiment, the FFT algorithm that is used needs to be able to do calculations on data of arbitrary length. The Bluestein's FFT algorithm, also known as the chirp z-transform algorithm, is an FFT algorithm that computes the discrete Fourier transform of arbitrary sizes by re-expressing the discrete Fourier transform as a convolution (Agarwal *et al.*, 1994). The Bluestein algorithm is therefore used to apply the FFT analysis to the accelerometer data. Figure 5-8, the result of applying the FFT analysis on the first gesture, up and down, in the frequency domain is presented. On the *y*-axis is the magnitude and on the *x*-axis are the three FFT terms of this specific gesture. The magnitude shows the strength in terms of the frequency when the FFT values are compared to each other. The *x*-axis is shown in red, the *y*-axis in green, and the *z*-axis in blue. One hundred iterations of each axis are shown in Figure 5-8. When gesture 1 is performed, the expected FFT values should be in the same ranges, as shown in Figure 5-8. There is some variation in the magnitude which could be caused by inconsistencies when the gesture is performed. For example, the speed at which the gesture is performed could influence the magnitude.

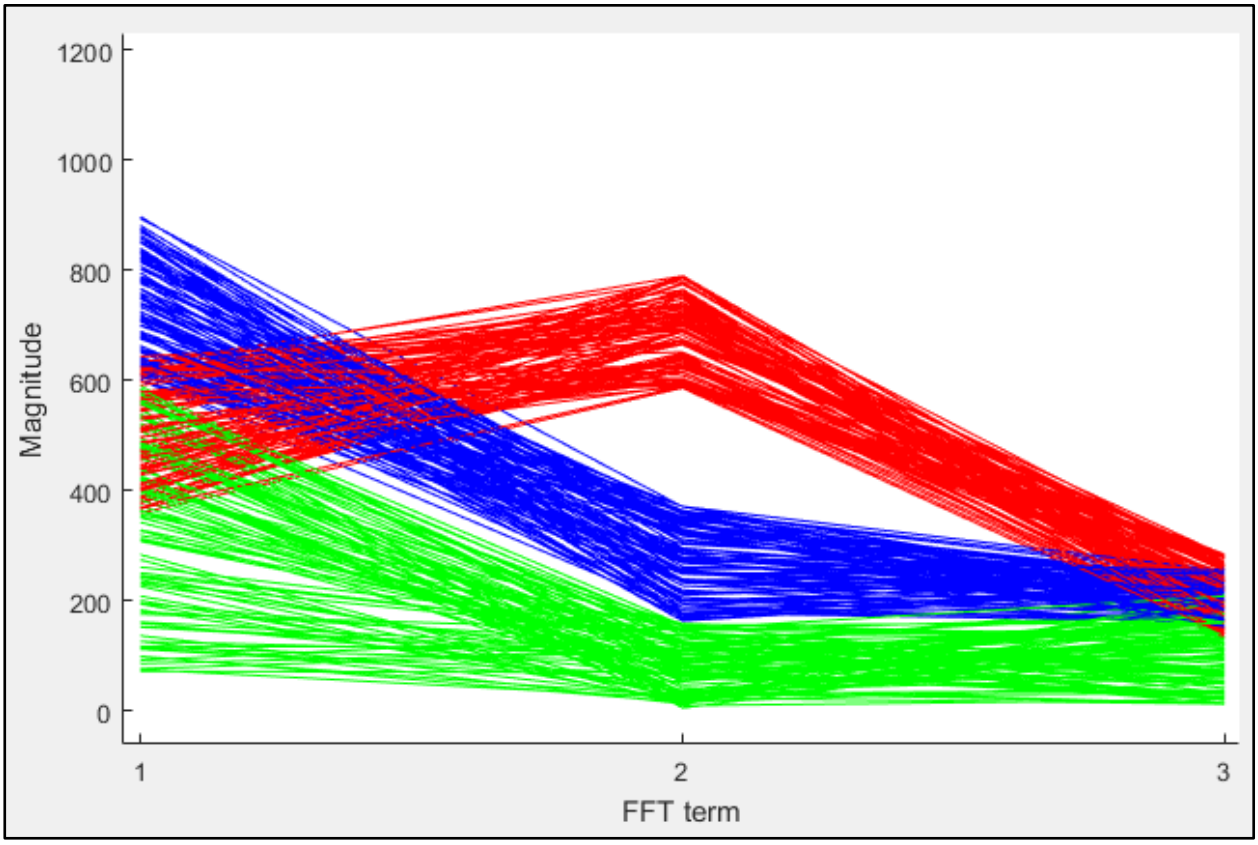


Figure 5-8: Results from the FFT analysis of gesture 1 (up and down)

The results from an FFT analysis for the second gesture, the circular motion, are presented in Figure 5-9 in the frequency domain.

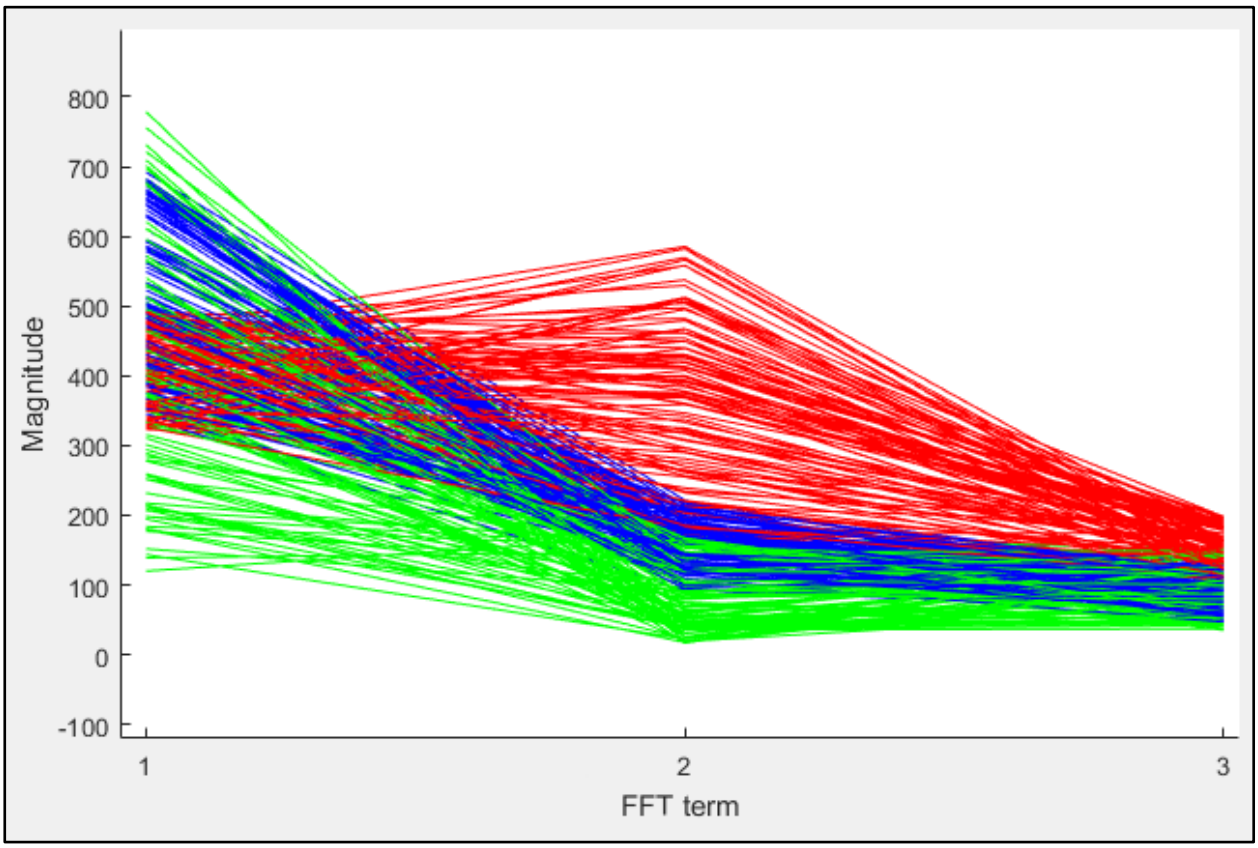


Figure 5-9: Results from the FFT analysis of gesture 2 (circle)

When the second gesture is performed, the FFT values should be relatively in the same ranges, as shown in Figure 5-9.

In Figure 5-10, the results from the third gesture, up, right, and left are presented in the frequency domain. When gesture 3 is performed, the expected FFT values should be in the same ranges, as shown in Figure 5-10. There is also some variation in the magnitude. This could also have been caused by the speed at which the gesture was performed. Getting these inconsistencies should be expected. The pilot will rarely perform the gesture in the same way. The speed might differ, or for example, the circle of gesture 2 might be smaller or bigger each time. As long as these inconsistencies are relatively small, it should be possible to get more or less the same result when the gesture is performed again.

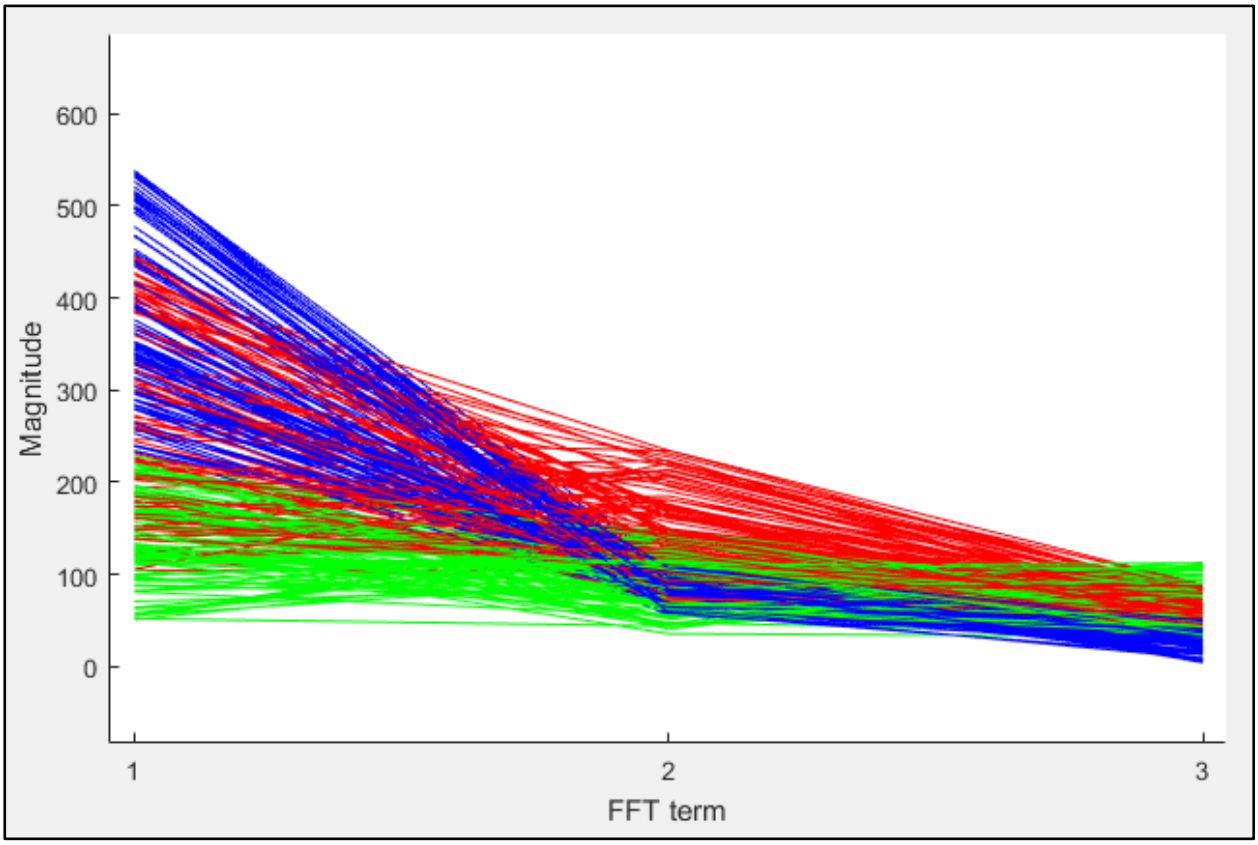


Figure 5-10: Results from FFT analysis of gesture 3 (up, right and left)

Figure 5-11 denotes the FFT values of all the gestures together.

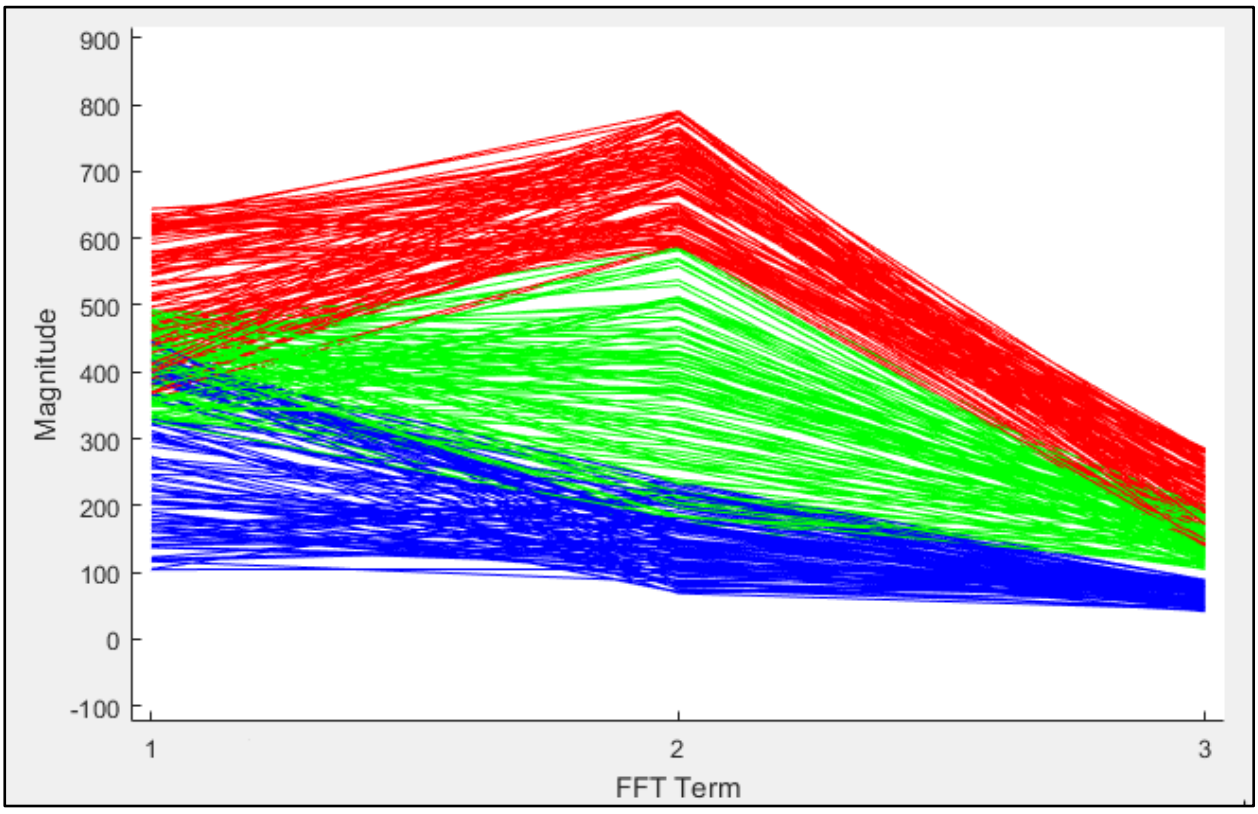


Figure 5-11: Comparing three gestures with a fast Fourier transform analysis

The first gesture is shown in red, and clusters at the top of the graph. The second gesture, shown in green, forms a group in the middle. The last gesture is indicated in blue and groups at the bottom of the graph. Visually, it is possible to distinguish between the different values by only using the first three FFT values of each gesture. The results from using an RBFNN for gesture recognition with the FFT data is presented and discussed in the next section.

5.3.3 RBFNN gesture recognition results

In Chapter 4, it was mentioned that the RBFNN was trained by using seventy per cent of the data, and twenty per cent was used to validate the model. The last ten per cent of the data was used to test the RBFNN. In Table 5-1 is a summary of the results that were obtained after testing the RFBNN model with the test data.

Result	Value
Correctly classified instances	96.67%
Incorrectly classified instances	3.33%
Kappa statistic	0.9667

Table 5-1: Result from testing the RBFNN model

The RBFNN correctly classified 96.67 per cent of the gestures, with only 3.33 per cent incorrectly classified. The kappa statistic was 0.9667. The kappa statistic is a measure of how closely the instances classified by the machine learning classifier matched the truth (Witten *et al*., 2011). The kappa coefficient is interpreted, using the guidelines outlined by Landis and Koch (1977). The strength of the kappa coefficients is interpreted, as shown in Table 5-2.

Value	Result
0.01 - 0.20	slight
0.21 - 0.40	fair
0.41 - 0.60	moderate
0.61 - 0.80	substantial
0.81 - 0.99	almost perfect
1.0	perfect

Table 5-2: Strength of the kappa coefficients

According to the guidelines, the kappa statistic of the model is almost perfect. This indicates that the RBFNN can successfully predict the correct gesture, based on the FFT values.

The confusion matrix is used for summarising the performance of a classification algorithm (Papapetrou *et al.*, 2011). By using the classification accuracy alone, can be misleading when the number of observations in each class is unequal or when there are more than two classes in the data set that was used for training. The confusion matrix gives a better idea of what the

classification model classifies correctly and what type of error it is making. Table 5-3 indicates the results of training the RBFNN for gesture recognition presented in the confusion matrix format. In the first column, are the actual gestures are depicted. In the first row, the last three columns display the gesture that was identified by the RBFNN. From the confusion matrix, it can be seen that there was one instance where the wrong classification was made. In the third column, the value 1 indicates that the classifier predicted it was gesture 2, but it should have been gesture 1.

	Predicted gesture		
Actual gesture	1	2	3
1	9	1	0
2	0	10	0
3	0	0	10

Table 5-3: Confusion matrix from RBFNN gesture recognition results

In the next section is a discussion on the results from testing the trained RBFNN model for gesture recognition in the project application on the smartphone.

5.3.3.1 Testing the RBFNN model in the project application

The previous results were obtained from Weka running on a desktop computer, mentioned in Chapter 4. The trained RBFNN model was also tested on the mobile project application to ensure the same results were obtained on the mobile application. Table 5-4 shows a sample of the test data that was used for testing the RBFNN model.

Gesture	X-1	X-2	X-3	Y-1	Y-2	Y-3	Z-1	Z-2	Z-3	Prediction
1	472.89	790.59	225.79	561.19	11.62	211.55	715.35	238.31	245.59	1
2	481.83	449.07	197.96	353.97	65.08	88.19	467.87	199.53	106.61	2
3	307.42	68.83	44.79	127.69	35.73	32.52	232.48	71.11	45.01	3
1	506.59	623.36	236.15	461.44	59.92	139.84	601.36	182.96	188.44	1
2	420.96	410.94	189.35	294.16	65.82	58.39	359.73	161.84	87.76	2
3	446.21	111.43	41.67	168.95	82.83	112.82	392.30	100.20	14.05	3
1	644.54	666.96	284.99	423.35	59.95	135.99	614.46	271.95	179.07	1
2	433.75	430.42	191.93	360.91	46.75	79.38	405.97	170.11	132.71	2
3	244.40	115.09	88.48	109.51	149.34	84.35	393.62	88.07	50.57	3

Table 5-4: Sample of test results from RBFNN gesture recognition

In the first column, the actual gesture is shown and in the last column is the gesture predicted by the RBFNN classifier. In Figures 5-12, 5-13, and 5-14 are examples of each gesture that was tested on the mobile application by using the test data in Table 5-4.

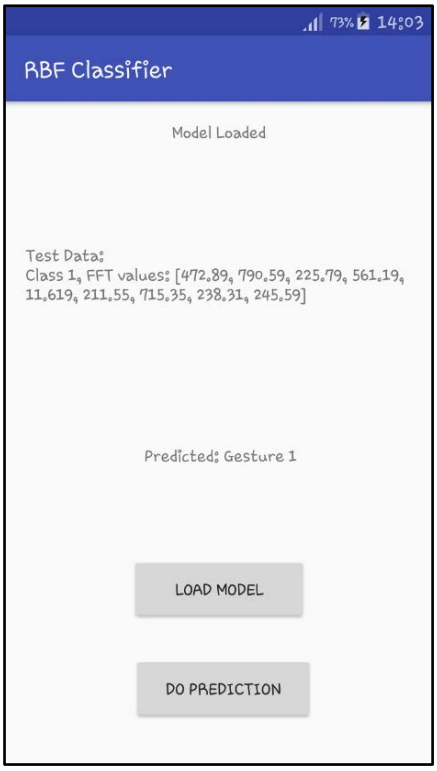


Figure 5-12: RBFNN gesture recognition results on the smartphone for gesture 1

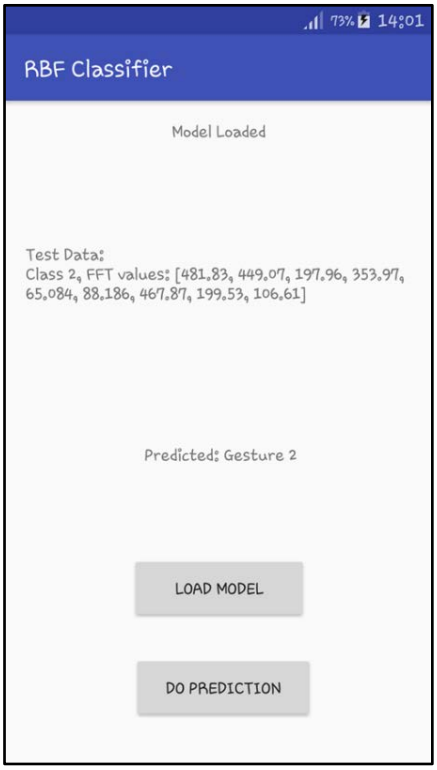


Figure 5-13: RBFNN gesture recognition results on the smartphone for gesture 2

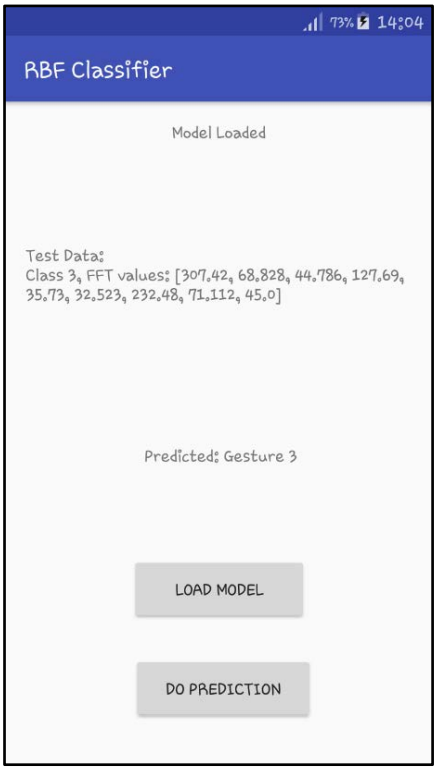


Figure 5-14: RBFNN gesture recognition results on the smartphone for gesture 3

From the cases presented in Figures 5-12, 5-13, and 5-14 it can be seen that the same results are obtained on the mobile application when comparing the data on the test data label and predictions label to the test data in Table 5-4. Only the results from the first three rows from Table 5-4 are presented. The results on the full test data set are available in Annexure B.

Most high-level piloting is controlled autonomously, but some tasks still need to be controlled manually. Take-off and landing are controlled autonomously, but still need to be initiated by pressing the take-off or landing button on the smartwatch. The UAS which follows the pilot is autonomously based on the user's GPS. The pilot does not have to control the UAS manually for the UAS to be able to follow the pilot. The following function just needs to be initiated by performing the gesture associated with the following function. The speed of the UAS needs to be adjusted manually by the hand-held device. Based on the automation level model presented in Chapter 3, this implementation uses level 2 automation. In the next section is a discussion on the activity recognition implementation results.

5.4 Improved UAS control with automated activity recognition results

In the previous implementation, gestures were identified by mapping accelerometer data to the three gestures. In this implementation, activities were used to map the accelerometer data to different activities. The results from the activity data should indicate that the different activities

can be differentiated from each other. The results from calculating the magnitude of the accelerometer data are discussed next.

5.4.1 Activity recognition accelerometer data magnitude results

The accelerometer data of all three axes for the activity recognition implementation was transformed into a single scalar value, as discussed in the previous chapter. Figure 5-15 is a visual representation of ten acceleration sample values per activity.

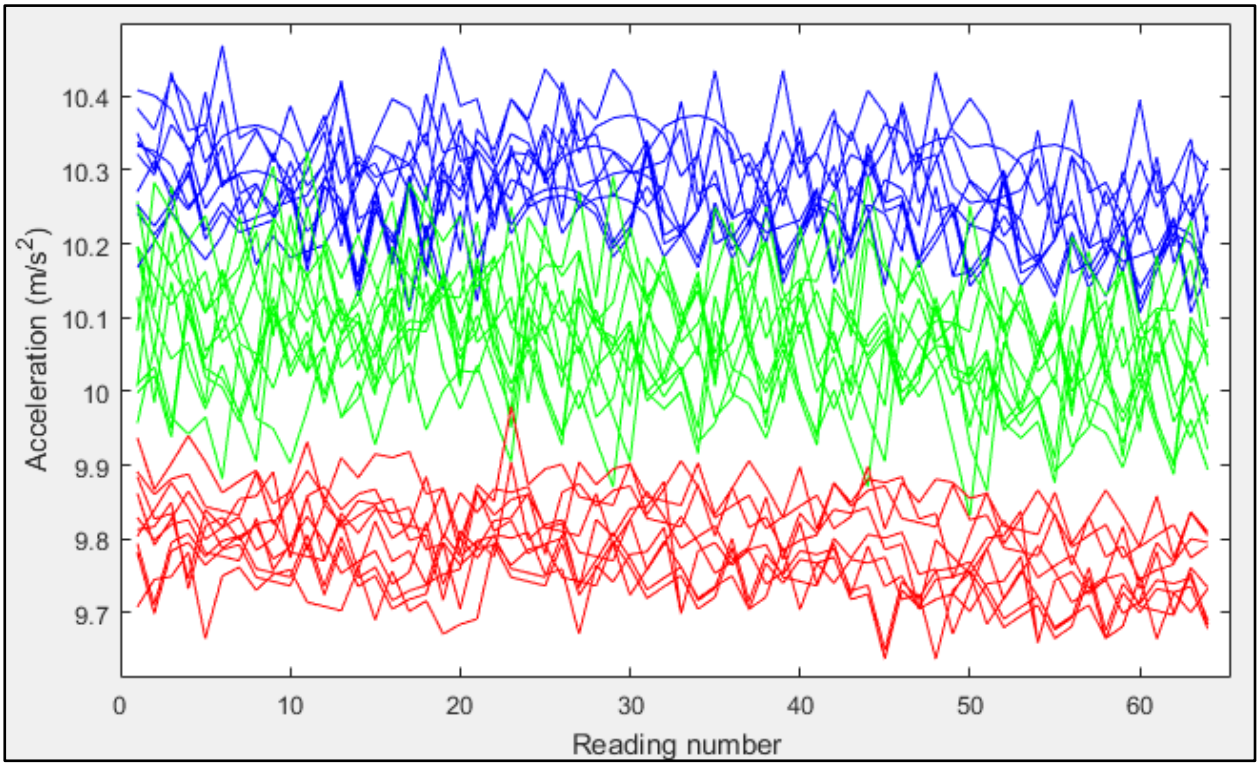


Figure 5-15: Results from calculating magnitude for accelerometer data

The standing activity is shown in red. Walking is shown in green, and running is shown in blue. From Figure 5-15, it can be seen that the acceleration of standing is the lowest, walking has the second-lowest and running has the biggest acceleration. The result from applying an FFT analysis on the acceleration data for activity recognition is discussed in the next section.

5.4.2 Activity recognition FFT accelerometer data results

Applying an FFT analysis to the accelerometer data for gesture recognition has proven to be very beneficial. An FFT analysis is also applied to the accelerometer data of the different activities to get more information about the data. The accelerometer data collected for the activity recognition experiment had the same data lengths (number of readings). Each iteration for each activity had sixty-four readings. It is therefore not necessary to use an FFT algorithm that can do calculations on data of arbitrary length as with the previous experiment. It is better

to use an FFT that can be a power of two (the number of accelerometer readings need to be a power of two) because it improves the performance of the calculations (Kerr, 1998). In Figure 5-16, the results from applying an FFT analysis on the first activity (standing) in the frequency domain is presented. One hundred iterations of the standing activity are presented. On the y-axis is the magnitude and on the x-axis are the FFT terms of this specific gesture. The magnitude indicates the strength of the accelerometer data after an FFT analysis was applied relative to each other. From the results, it can be seen that there is some variation in the results even though this is the standing activity. This can be caused by the pilot moving his/her arm while standing still. When performing the standing activity, it is expected that the results will be in the same ranges as presented in the figure.

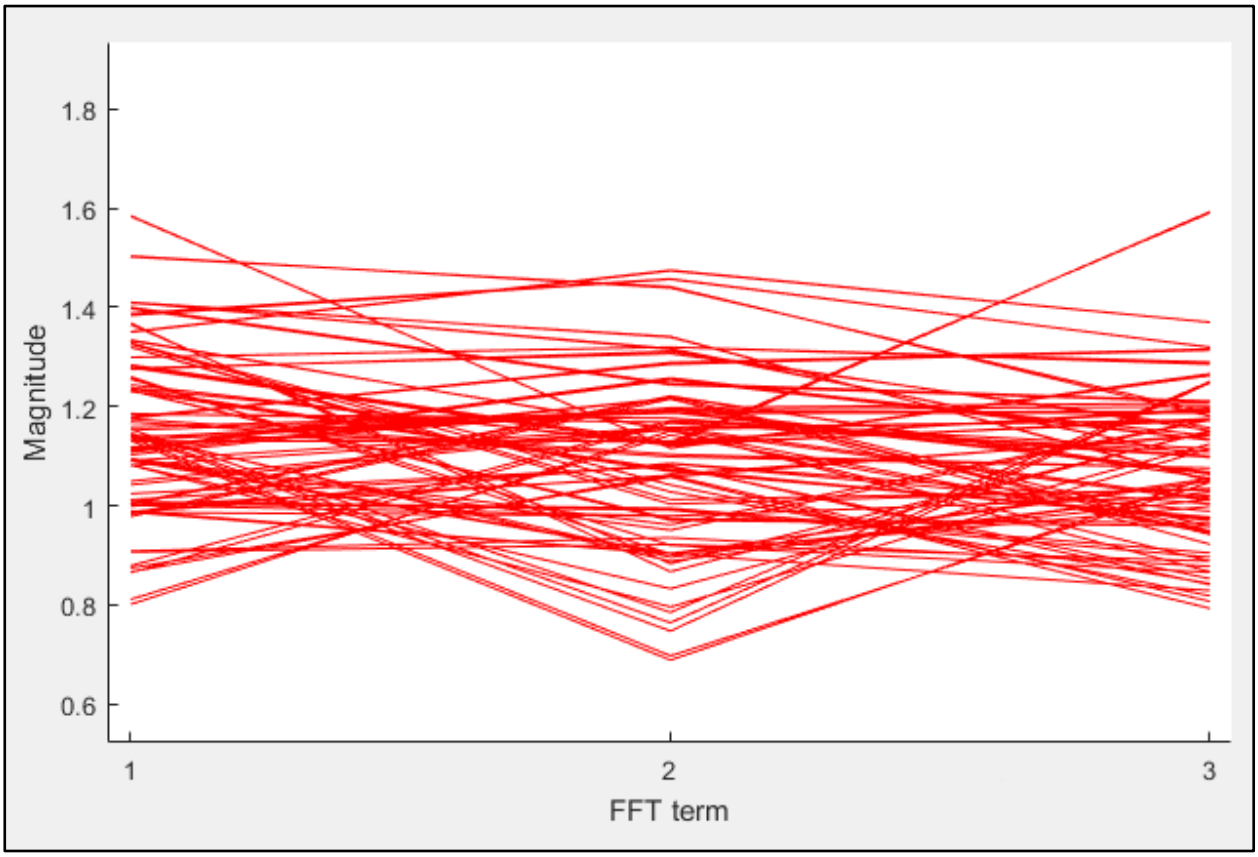


Figure 5-16: Results from FFT analysis on the first activity (standing)

The results from applying an FFT analysis on the accelerometer data from the second activity (walking) in the frequency domain for one hundred iterations are presented in Figure 5-17. The variations in the results can be caused by the different speeds at which the pilot is walking.

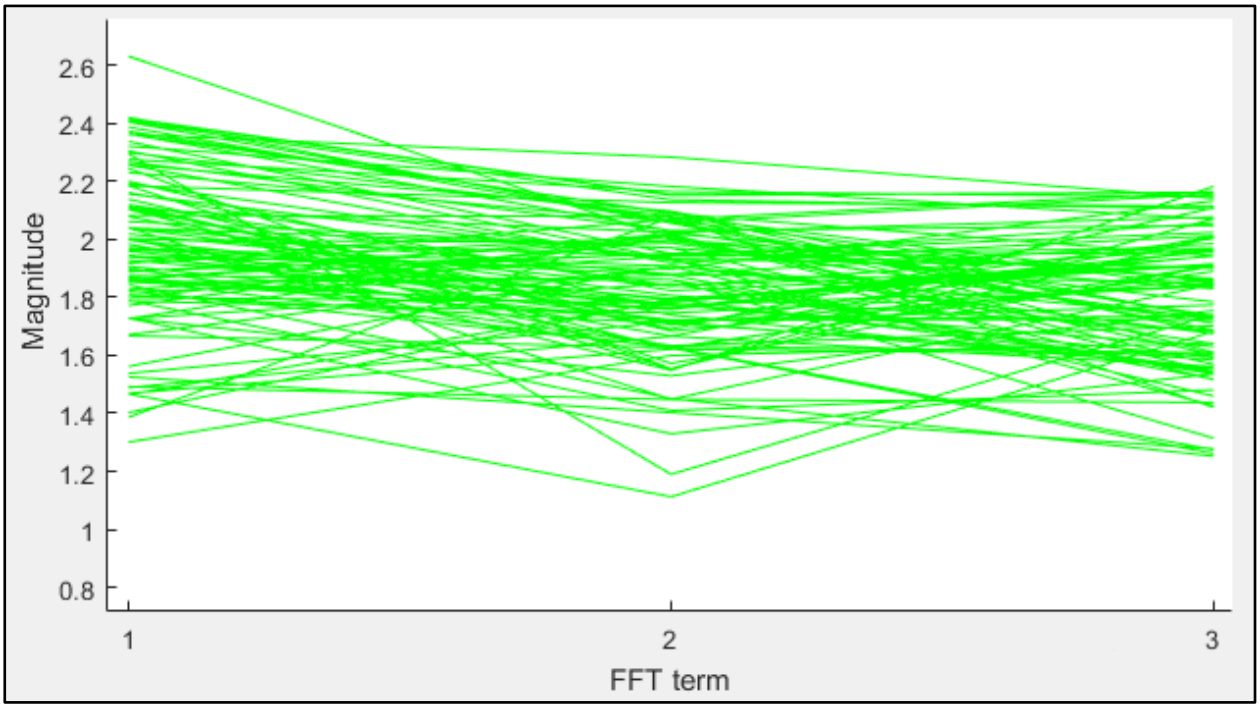


Figure 5-17: Results from FFT analysis on the second activity (walking)

In Figure 5-18, the results from applying an FFT analysis on the third activity (running) in the frequency domain are presented for one hundred iterations.

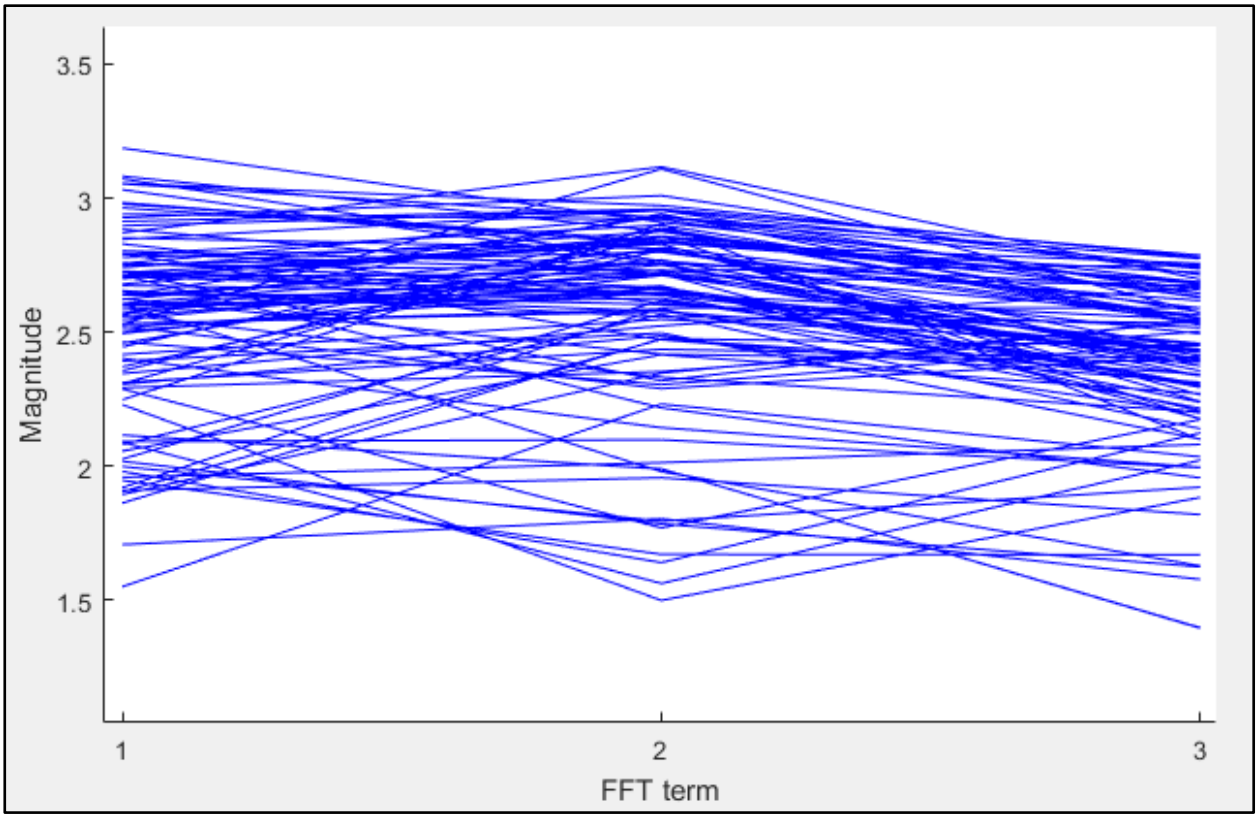


Figure 5-18: Results from FFT analysis on the third activity (running)

There are also some variations in the results which could be caused by the different speeds at which the pilot was running.

Figure 5-19 depicts the results of Figures 5-16, 5-17, and 5-18 combined. From Figure 5-19, it can be seen that the standing activity is shown in red with the lowest magnitude. This is as expected, since the pilot carried out little to no movement. The walking activity shown in green has a higher magnitude than the standing activity. The blue lines represent the running activity. The running activity shows the highest magnitude on average.

From the results shown by the FFT analysis, it is visually possible to differentiate between the different activities.

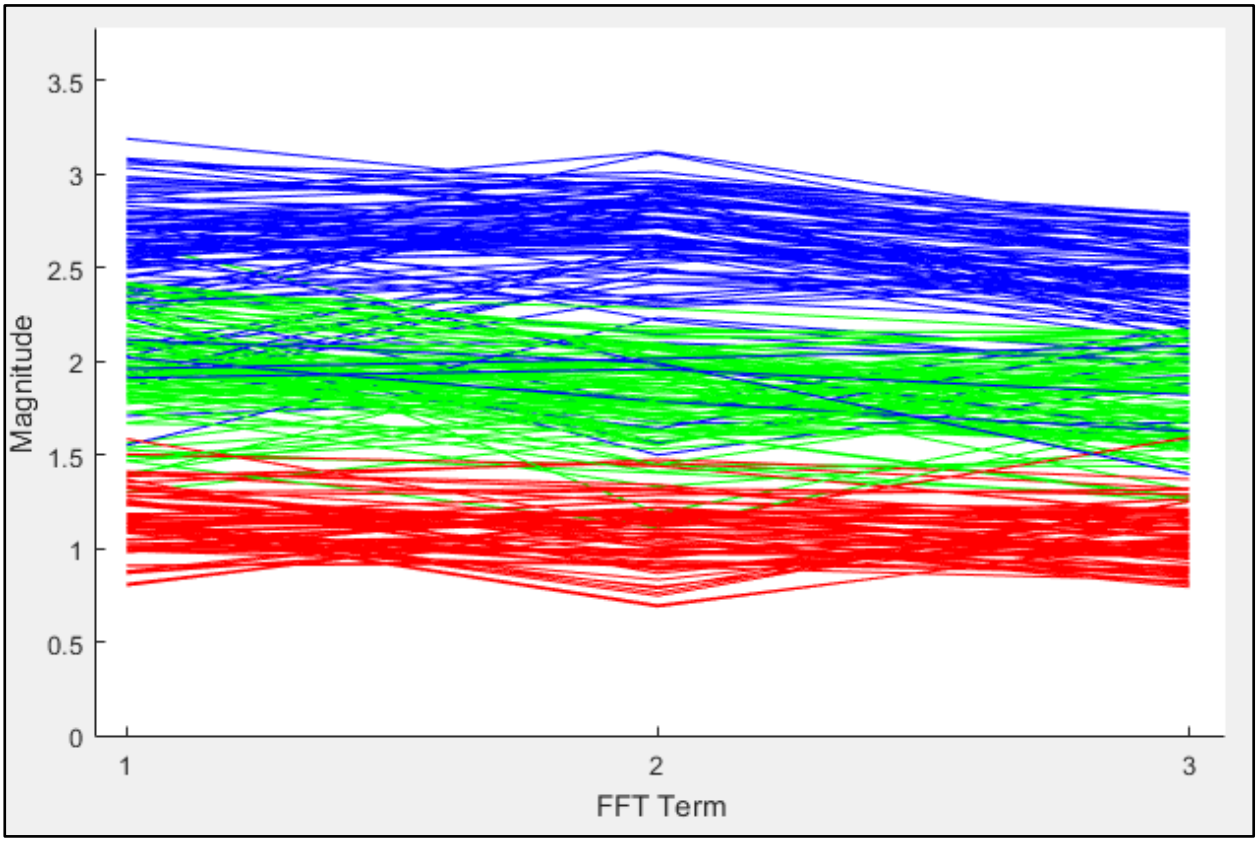


Figure 5-19: Results from converting accelerometer data into FFT values for all three activities

In the next section is a discussion of the results obtained from implementing activity recognition with an RBFNN.

5.4.3 RBFNN activity recognition results

Seventy per cent of the data was used to train the model, and twenty per cent was used to validate the model. The last ten per cent of the data was used to test the model. The test results are shown in Table 5-5.

Result	Value
Correctly classified instances	93.33%
Incorrectly classified instances	6.67%
Kappa statistic	0.9333

Table 5-5: Result from testing the RBFNN model

The RBFNN correctly classified 93.33 per cent of the gestures with 6.67 per cent incorrectly classified. According to the kappa statistic guidelines shown in Table 5-2, the model is almost perfect.

The results from testing the activity recognition model are also shown as a confusion matrix in Table 5-6. From the table, it can be seen that all the standing activities were correctly predicted for this test data set. For the second activity, the walking activity, there is one case where the classifier predicted that the pilot was currently running instead of walking. This also happened for the running activity, but in this case, the classifier predicted that the pilot was walking instead of running

	Predicted activity		
Actual activity	1	2	3
1	10	0	0
2	0	9	1
3	0	1	9

Table 5-6: Results from RBFNN gesture recognition

The results show that the RBFNN can successfully predict the current activity performed by the pilot to enable autonomous UAS speed control. In Section 5.3.3.1, the results from Weka on the desktop computer were compared to the results obtained on the mobile platform to ensure that the same results were obtained on both platforms. The Weka results from the desktop computer were also compared for the activity recognition implementation and can be seen in Annexure B.

The activity recognition system was implemented to further improve and automate UAS control by automating the speed of the UAS. The UAS follows the pilot and adjusts the speed

autonomously. Most of the UAS control is now autonomously controlled. Based on the automation level model discussed in Chapter 3, this implementation uses level 3 automation.

Table 5-7 is a summary of how the experiments improved UAS control and improved automation.

Experiment	Control simplification	Automation improvement
Basic control implementation	With the basic control system, the UAS is mostly remotely piloted except for take-off, landing, and emergency controls. There are no follow capabilities built into the basic control system. The UAS needs to be manually controlled to be able to follow the pilot and the speed of the UAS also needs to be updated manually. No control simplification is established.	This implementation is used as the base for comparison, and no improvements were made to automation. Based on the levels of UAS automation, this system uses level 1-automation.
Smartwatch accelerometer data control	Controlling the UAS with smartwatch accelerometer data enables the pilot to control the UAS without the need to have a smartphone in his/her hand.	Automation is not improved with this implementation, and the level of automation for this system is also level 1.
Gesture recognition	The gesture recognition system also enables the pilot to control the UAS without having the smartphone in his/her hand. Unlike the previous implementation, the UAS does not move when the pilot moves his/her arm because the accelerometer data is not used to control the UAS directly.	Gesture recognition is used for control instructions which enable most high-level piloting to be controlled autonomously. Based on the levels of UAS automation, this implementation uses level 2-automation.
Follow Function	The pilot does not have to control the UAS manually for the UAS to be able to follow the pilot.	The follow function enables the UAS to follow the pilot autonomously. Although this implementation improves automation, based on the levels of UAS automation, this system also uses level 2-automation.
Activity recognition	Activity recognition enables the speed of the UAS to be autonomously adjusted, and there is no need for the pilot to set the speed manually.	Most of the UAS control is now autonomously controlled. Based on the UAS automation level model, this implementation uses level 3-automation.

Table 5-7: Summary of control simplification and automation improvements

In the next section, a conclusion to the chapter is presented.

5.5 Conclusion

The results from the various systems implemented in Chapter 4 were presented and discussed in this chapter. These results are used to provide evidence that UAS control can be simplified and that automation can be improved by using gesture and activity recognition with an RBFNN. The basic control system results showed that the system uses little to no automation, except for take-off, landing and the emergency piloting control task. Results from the second implementation indicated that it is possible to control the UAS with the smartwatch acceleration data, but it did not improve UAS automation. Gesture control was used in the third implementation, and the results showed that it is possible to use accelerometer data with an RBFNN to improve and automate UAS control. The last implementation results showed that it is possible to improve UAS control and automation by automating the speed of the UAS with the help of an RBFNN. In the final chapter, the study is concluded.

CHAPTER 6 CONCLUSION

UASs have a rich history that goes back to ancient times. Throughout the years, the UAS has become both lightweight and cost-effective. UASs have tended to be driven by military applications, but in the last few years, various commercial fields have also been utilising the UASs (Chapter 1). The study aimed to simplify UAS control and improve automation, using an RBFNN for hand gesture and activity recognition. There are currently various methods to control UASs (Chapter 3). It is also notable that many automation techniques are also being investigated and improved to simplify UAS control. In this study, some of these techniques were explored and improved to simplify UAS control.

The remainder of the chapter will be as follows. The research goals are evaluated in Section 6.1. In Section 6.2, is a summary of the key contributions from this study. Future work is discussed in Section 6.3, followed by the conclusion of the chapter in Section 6.4.

6.1 Evaluation of research goals

The research question for this study was presented in Chapter 1, Section 1.6 as follows: “Can hand gestures and activities that are classified by a radial basis function neural network (RBFNN) be used to simplify UAS control and improve UAS automation?” To address this research question, seven secondary objectives were set. A discussion of how these secondary objectives were met is presented below.

1. **Perform a literature review on UAVs, UASs, applications of UASs, the positivistic research paradigm, the design science research strategy, the SCRUM development methodology, control systems (manual and autonomous), automation of UAS control techniques, artificial neural networks, RBFNNs and the FFT algorithm.**

In Chapter 1, literature on UAVs and UASs was discussed. Various terms and acronyms that are used in the field of UAVs were considered. Some of the most prominent applications from the literature were briefly mentioned to illustrate the pervasive role UASs play in the public domain. The positivistic research paradigm, the design science research strategy and the SCRUM development methodology were discussed in Chapter 2. In Chapter 3, control systems (manual and autonomous), automation of UAS control techniques, artificial neural networks, RBFNNs and the FFT algorithm were considered.

2. **Implement a basic UAS control system which will act as the starting point for comparisons.**

In Chapter 4, Section 4.1, a basic UAS control system was implemented. The system used a traditional UAS control architecture. The UAS was piloted by means of remote control and had

only basic functionality. Only the take-off, land, and emergency functions were autonomously controlled. Key findings for the basic control system were discussed in Chapter 5, Section 5.1.

3. **Improve the system in (2) by controlling the UAS with smartwatch accelerometer data.**

The third objective was achieved by implementing a UAS control system that used accelerometer data from a smartwatch, as discussed in Chapter 4, Section 4.2. Pitch, yaw, and roll movements obtained from accelerometer data were used to control the UAS. The results were discussed in Chapter 5, Section 5.2. This implementation improved the system in (2) by enabling the pilot to control the UAS without the need to have the smartphone in his/her hand.

4. **Enhance the system in (3) with gesture recognition implemented through an RBFNN.**

Three gestures were identified in Chapter 4, Section 4.3.1. These gestures were used for UAS control. Accelerometer data for each gesture were recorded and mapped to various control tasks that could be performed by the UAS. The gesture recognition system enhanced the system in (3) by using gesture recognition with an RBFNN instead of using accelerometer data directly to control the UAS. The results were discussed in Chapter 5, Section 5.3.

5. **Extend system (4) by implementing a pilot follow function to simplify UAS control.**

In Chapter 4, Section 4.4, a follow function was implemented to enable the UAS to follow the pilot with the help of GPS information autonomously. The follow function simplified UAS control by enabling the UAS to follow the pilot autonomously. There was no need for the pilot to use a smartphone to control the UAS to follow him/her continuously.

6. **Improve UAS control of (5) with automated activity recognition using an RBFNN.**

An automated activity recognition system was implemented in Chapter 4, Section 4.5. In this implementation, the accelerometer data were mapped to different activities with the help of an RBFNN. The activity recognition system further improved and automated UAS control by automating the speed of the UAS. The UAS followed the pilot and adjusted the speed autonomously. The results were discussed in Chapter 5, Section 5.4.

7. **Evaluate the implemented systems, based on the simplification of control and the improved levels of UAS automation.**

Each system was evaluated in Chapter 5 based on the simplification of UAS control and the four levels of UAS automation presented in Chapter 3, Section 3.3.4. In Chapter 5, Table 5-7 is a summary of the evaluations.

All the goals set in Chapter 1 were met, and the following key findings were made:

- **Literature review:** UASs are used in various commercial and military fields. There are currently various control methods that can be used to control a UAS. From the research, it is evident that there are numerous control and automation techniques that can be used to simplify and automate UAS control.

- **Smartwatch control:** The UAS can be controlled with a smartwatch by using the accelerometer data provided by the smartwatch when the pilot moves his/her hand. The downside to this implementation is that the UAS moves whenever the pilot moves his/her hand. The UAS also moves when the pilot is not standing still. This makes it difficult to control the UAS while the pilot is moving.

- **Gesture recognition:** Gestures can be mapped to the accelerometer data gathered from the smartwatch. The process of applying an FFT analysis to the data provides more information about the accelerometer data. An RBFNN can classify the gesture that was performed by the pilot, based on these FFT values with high accuracy. These gestures can be assigned to various control tasks that can be performed by the UAS.

- **UAS following function:** The GPS coordinates of the pilot's hand-held device can be used to instruct the UAS to navigate to the provided GPS coordinates.

- **Activity recognition:** Activities can be mapped to accelerometer data gathered from the smartwatch while the pilot is flying the UAS. Applying an FFT analysis also provided more information about the accelerometer data. An RBFNN can also be trained and used to classify the activity, based on the FFT values. The speed of the UAS can then be autonomously updated, based on the activity classification made by the RBFNN.

In the next section is a summary of the contributions made in this study.

6.2 Summary of contributions

This study made the following contributions:

- A literature review on UAVs, UASs, applications of UASs, the positivistic research paradigm, the design science research strategy, the SCRUM development methodology, control systems (manual and autonomous), automation of UAS control techniques, artificial neural networks, RBFNNs and the FFT algorithm

- The sample application provided by the manufacturer of the Parrot for the Bebop Drone was enhanced to enable the smartphone to communicate with the smartwatch in order the gather accelerometer data from the smartwatch which enabled the UAS to be piloted, based on the accelerometer data.
- A gesture recognition system was implemented, enabling the UAS control task to be mapped to gestures with the help of an RBFNN.
- A follow function was implemented to enable the UAS to follow the pilot based on his/her GPS coordinates autonomously.
- An activity recognition system was implemented to enable the UAS to autonomously update the speed of the UAS, based on the current activity performed by the pilot with the help of an RBFNN.
- Results from the experiments that were implemented to simplify UAS control and improve automation were obtained and examined.

Suggestions for future work are discussed next.

6.3 Suggestions for future work

Further research could focus on implementing autonomous UAS control with the aid of vision control. In Chapter 3, it was mentioned that various vision-based techniques could be used to enable the UAS to follow the pilot. The results can be compared to the GPS implementation of this study to determine which implementation works better or to determine if the two implementations can be combined to minimise the disadvantages that may be presented by each implementation.

A utility-based agent could be implemented for the UAS. The agent could be given some goals (UAS control tasks) to perform, and the RBFNN could then evaluate the state according to the user's feedback. Through trial and error, the agent could learn which control task the UAS should autonomously perform in a similar scenario.

6.4 Conclusion

In this study, the research question "Can hand gestures and activities that are classified by a radial basis function neural network (RBFNN) be used to simplify UAS control and improve UAS automation?" was investigated. The results in Chapter 5 showed that an RBFNN could accurately classify hand gestures and activities performed by the pilot based on the accelerometer data recorded from a smartwatch to simplify UAS control and improve automation.

BIBLIOGRAPHY

Agarwal, R.C., Gustavson, F.G., & Zubair, M. 1994. A high performance parallel algorithm for 1-D FFT. *In*: Proceedings of Supercomputing (ACM). pp. 34-40.

Altu, E., Ostrowski, J.P., & Mahony, R. 2002. Control of a Quadrotor Helicopter Using Visual Feedback. *In*: International Conference on Robotics and Automation, Washington, DC, USA. pp. 72-77.

American Kitefliers Association. 2016. History of Kites. URL http://kite.org/education/history-of-kites/. Date of access: 3 Apr. 2019.

Anand, S.S. & Mathiyazaghan, R. 2016. Design and Fabrication of Voice Controlled Unmanned Aerial Vehicle. *Journal of Aeronautics & Aerospace Engineering*. 5(2):1-5.

Android Studio. 2017. Android Developers. URL https://developer.android.com/studio/index.html. Date of access: 15 Aug. 2017.

Austin, R. 2010. *Unmanned aircraft systems: UAVS design, development and deployment.* 54th edition. John Wiley & Sons. pp. 505-531.

Bachrach, A. 2009. Autonomous Flight in Unstructured and Unknown Indoor Environments. *International Journal of Micro Air Vehicles.* 1(4):126.

Baddeley, A.D. & Hitch, G. 1974. Working memory. *Psychology of Learning and Motivation.* 8:47-89.

Bainbridge, L. 1983. Ironies of automation. *In Analysis, design and evaluation of man-machine systems.* 19(6):129-135.

Bartak, R. & Vykovsky, A. 2016. Any object tracking and following by a flying drone. *In*: 14th Mexican International Conference on Artificial Intelligence: Advances in Artificial Intelligence (MICAI), Cuernavaca, Mexico. pp. 35-41.

Beard, R.W., McLain, T.W., Nelson, D.B., Kingston, D., & Johanson, D. 2006. Decentralized cooperative aerial surveillance using fixed-wing miniature UAVs. *In*: Proceedings of the IEEE. 94(7):1306-1323.

Becerra, V.M. 2019. Autonomous control of unmanned aerial vehicles. *Electronics*. 8(4):1-4.

Begel, A. & Nagappan, N. 2007. Usage and perceptions of Agile software development in an industrial context: An exploratory study. *In*: First International Symposium on Empirical Software Engineering and Measurement (ESEM), Madrid, Spain. pp. 255-264.

Benjamin, M.R., Newman, P., Leonard, J., & Schmidt, H. 2009. A Tour of MOOS-IvP Autonomy Software Modules. MIT Internal.

Bennett, K.B. & Flach, J.M. 1992. Graphical Displays: Implications for Divided Attention, Focused Attention, and Problem Solving. *Human Factors: The Journal of the Human Factors and Ergonomics Society.* 34(5):513-533.

Berclaz, J., Fleuret, F., Türetken, E., & Fua, P. 2011. Multiple object tracking using k-shortest paths optimization*. IEEE Transactions on Pattern Analysis and Machine Intelligence*. 33(9):1806-1819.

Bhardwaj, A., Sam, L., Akanksha, Martín-Torres, F.J., & Kumar, R. 2016. UAVs as remote sensing platform in glaciology: Present applications and future prospects. *Remote Sensing of Environment*. *Elsevier Inc*. 175:196-204.

Bills, C., Chen, J., & Saxena, A. 2011. Autonomous MAV Flight in Indoor Environments using Single Image Perspective Cues. *In*: International Conference on Robotics and Automation, Shanghai, China: IEEE. pp. 5776-5783.

Blankenship, D.C. 2010. Applied Research and Evaluation Methods in Recreation. Human Kinetics. pp. 200

Bouabdallah, S., Murrieri, P., & Siegwart, R. 2004. Design and control of an indoor micro quadrotor. *In*: International Conference on Robotics and Automation (ICRA), New Orleans, LA, USA: IEEE. pp. 4393-4398.

Broomhead, D.S. & Lowe, D. 1988. Multivariable functional interpolation and adaptive networks. *Complex Systems*. 3(4):579-588.

Bry, A., Bachrach, A., & Roy, N. 2012. State estimation for aggressive flight in GPS-denied environments using on-board sensing. *In*: International Conference on Robotics and Automation, Saint Paul, MN, USA: IEEE. pp. 1-8.

Bujari, A., Licar, B., & Palazzi, C.E. 2012. Movement pattern recognition through smartphone's accelerometer. *In*: Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA: IEEE. pp. 502-506.

Cardozo, E., Araújo Neto, B., Barza, A., França, C., & da Silvia, F. 2010. SCRUM and Productivity in Software Projects : A Systematic Literature Review. *In*: International Conference on Evaluation and Assessment in Software Engineering (EASE). pp. 1-4.

Castrounis, A. 2016. Artificial Intelligence, Deep Learning, and Neural Networks Explained. URL http://www.innoarchitech.com/artificial-intelligence-deep-learning-neural-networks-explained/. Date of access: 5 Mar. 2017.

Çelik, K., Chung, S.J., Clausman, M., & Somani, A.K. 2009. Monocular vision SLAM for indoor aerial vehicles. *In*: International Conference on Intelligent Robots and Systems (IROS), St. Louis, MO, USA: IEEE. pp.1566-1573.

Chakrabarty, A., Morris, R., Bouyssounouse, X., & Hunt, R. 2016. Autonomous indoor object tracking with the Parrot AR.Drone. *In*: International Conference on Unmanned Aircraft Systems (ICUAS), Arlington, VA, USA: IEEE. pp. 25-30.

Chandarana, M., Meszaros, E.L., Trujillo, A., & Allen, B.D. 2017. "Fly Like This": Natural Language Interfaces for UAV Mission Planning. *In*: The Tenth International Conference on Advances in Computer-Human Interactions (ACHI). pp. 40-46.

Chandarana, M., Trujillo, A., Shimada, K., & Danette, A. 2017. A natural interaction interface for UAVs using intuitive gesture recognition. *In*: Advances in Human Factors in Robots and Unmanned Systems. Springer: pp. 387-398.

Chao, H., Cao, Y., & Chen, Y. 2010. Autopilots for small unmanned aerial vehicles: A survey. *International Journal of Control, Automation and Systems*. 8(1):36-44.

Charara, S. 2015. Moto 360 second gen review. URL https://www.wareable.com/motorola/moto-360-2nd-generation-2015-review. Date of access: 6 Jun. 2017.

Checchin, P., Gérossier, F., Blanc, C., Chapuis, R., & Trassoudaine, L. 2010. Radar Scan Matching SLAM using the Fourier-Mellin Transform. *Field and Service Robotic*. 1:1-10.

Chen, H., Wang, X.M., & Li, Y. 2009. A survey of autonomous control for UAV. *In*: International Conference on Artificial Intelligence and Computational Intelligence (AICI), Shanghai, China: IEEE. pp. 267-271.

Chivasa, W., Mutanga, O., & Biradar, C. 2017. Application of remote sensing in estimating maize grain yield in heterogeneous African agricultural landscapes: A review. *International Journal of Remote Sensing*. 38(23):6816-6845.

Cho, A. 2016. How to fly a pipe. URL https://www.hobbytron.com/lc/how-to-fly-a-quadcopter.html. Date of access: 7 Nov. 2019.

Clarke, R. 2014. Understanding the drone epidemic. *Computer Law and Security Review. Elsevier Ltd*. 30(3):230-246.

Conte, G. & Doherty, P. 2009. Vision-Based Unmanned Aerial Vehicle Navigation Using Geo-Referenced Information. *Journal on Advances in Signal Processing*. 1-18.

Criado, R.M. & Rubio, F.R. 2015. Autonomous path tracking control design for a commercial quadcopter. *IFAC-PapersOnLine*. 28(9):73-78.

De Almeida Maia, H., De Oliveira, F.L.M., & Vieira, M.B. 2016. Independent selection and validation for tracking-learning-detection. *In*: International Conference on Image Processing (ICIP), Phoenix, AZ, USA: IEEE. pp. 3469-3473.

Degeling, H. (2010). *Data collection methods*. URL http://mypeer.org.au/monitoring-evaluation/data-collection-methods/. Date of access: 8 Aug. 2016.

Deng, L. & Yu, D. 2013. Deep Learning: Methods and Applications. *Foundations and trends in signal processing*. 7(3-4):197-387.

DJI. 2017. Spark series. URL https://www.dji.com/products/spark?site=brandsite&from=nav. Date of access: 8 Jan. 2018.

DJI. 2019. LiDAR Equipped UAVs. URL http://enterprise.dji.com/news/detail/how-lidar-is-revolutionizing-mapping-and-geospatial-data. Date of access: 9 Jul. 2017.

Dorf, R.C. & Bishop, R.H. 2011. Modern Control Systems. 12th ed., New Jersey: Pearson.

Draper, M., Calhoun, G., Ruff, H., Williamson, D., & Barry, T. 2003. Manual versus speech input for unmanned aerial vehicle control station operations. *In*: Proceedings of the Human Factors and Ergonomics Society, CA, Los Angeles: SAGE. pp. 109-113.

Drone Industry Insights. 2017. Drone Energy Sources - Pushing the Boundaries of Electric Flight. URL https://www.droneii.com/drone-energy-sources. Date of access: 5 Nov. 2019.

Dudovskiy, J. 2016. Positivism from Research methodology. URL http://research-methodology.net/research-philosophy/positivism/. Date of access: 3 Oct. 2016.

Durrie, J., Gerritsen, T., Frew, E.W., & Pledgie, S. 2009. Vision-aided inertial navigation on an uncertain map using a particle filter. *In*: International Conference on Robotics and Automation, Kobe, Japan: IEEE. pp. 4189-4194.

Dusha, D. & Mejias, L. 2010. Attitude Observability of a Loosely-Coupled GPS / Visual Odometry Integrated Navigation Filter. *In*: Australasian Conference on Robotics and Automation (ACRA). pp. 1-41.

Dyring, E. 1973. The principles of remote sensing. Ambio. 11(3):57–69.

Edirisinga, P. 2012. Interpretivism and Positivism (Ontological and Epistemological Perspectives). URL https://prabash78.wordpress.com/2012/03/14/interpretivism-and-postivism-ontological-and-epistemological-perspectives/. Date of access: 3 Aug. 2016.

Ellis, T.J. & Levy, Y. 2008. Framework of problem-based research: A guide for novice researchers on the development of a research-worthy problem*. Informing Science*. 11:17-33.

Engel, J., Sch, T., & Cremers, D. 2014. LSD-SLAM: Large-scale direct monocular SLAM. *In*: European Conference on Computer Vision. pp. 834-849.

European RPAS Steering Group, 2013. Roadmap for the integration of civil remotely-piloted aircraft systems into the European Aviation System. *Final report from the European RPAS Steering Group*.

Federal Aviation Administration. 2012. Balloon Flying Handbook. Skyhorse Publishing.

Federal Aviation Administration. 2008. Unmanned aircraft systems operations in the U. S. national airspace system. Available at: www.faa.gov/uas.

Feigl, H. 2015. *Positivism*.URL http://global.britannica.com/topic/positivism. Date of access: 3 Aug. 2016.

Fernandez, R.A.S., Sanchez-Lopez, J.L., Sampedro, C., Bavle, H., Molina, M., & Campoy, P. 2016. Natural user interfaces for human-drone multi-modal interaction. *In*: International Conference on Unmanned Aircraft Systems (ICUAS), Arlington, VA, USA: IEEE. p.p. 1013-1022.

Finn, R.L. & Wright, D. 2012. Unmanned aircraft systems: Surveillance, ethics and privacy in civil applications. *Computer Law and Security Review*. 28(2):184-194.

Frank, E., Hall, M.A., & Witten, I.H. 2016. The WEKA Workbench. *Data Mining: Practical Machine Learning Tools and Techniques.* 1:128.

Fraundorfer, F., Heng, L., Honegger, D., Lee, G.H., Meier, L., Tanskanen, P., & Pollefeys, M. 2012. Vision-based autonomous mapping and exploration using a quadrotor MAV. *In*: International Conference on Intelligent Robots and Systems, Vilamoura, Portugal: IEEE. pp. 4557-4564.

Gaszczak, A., Breckon, T.P., & Han, J. 2011. Real-time people and vehicle detection from UAV imagery. *Intelligent Robots and Computer Vision XXVIII: Algorithms and Techniques*. 7878:1-13.

Gechman, M. & Gechman, M. 2019. Software Development Methodologies. *In*: Project Management of Large Software-Intensive System. CRC Press: pp. 49-66.

Ghosh, D.K. & Ari, S. 2011. A static hand gesture recognition algorithm using k-mean based radial basis function neural network. *In*: International Conference on Information, Communications and Signal Processing (ICIS), Singapore: IEEE. pp. 1-5.

Giusti, A., Guzzi, J., Cire, D.C., He, F., Rodríguez, J.P., Fontana, F., Fässler, M., Forster, C., Schmidhuber, J., Caro, G. Di, Scaramuzza, D., & Gambardella, L.M. 2015. A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots. *IEEE Robotics and Automation Letters*. 1(2):661-667.

Grenzdörffer, G., Engel, a, & Teichert, B. 2008. The photogrammetric potential of low-cost UAVs in forestry and agriculture*. International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences.* 1:1207-1213.

Grewal, M.S., Weill, L.R., & Andrews, A.P. 2007. Global Positioning Systems, Inertial Navigation, and Integration. John Wiley & Son.

Güldenring, J., Gorczak, P., Eckermann, F., Patchou, M., Tiemann, J., Kurtz, F. and Wietfeld, C. 2020. Reliable Long-Range Multi-Link Communication for Unmanned Search and Rescue Aircraft Systems in Beyond Visual Line of Sight Operation. *Drones*. 4(2):16.

Haluza, M. & Cechak, J. 2016. Analysis and decoding of radio signals for remote control of drones. *In*: New Trends in Signal Processing (NTSP), Demanovska Dolina, Slovakia: IEEE. pp. 1-5.

Hanlon, M. 2008. ScanEagle UAV gets Synthetic Aperture Radar. URL http://www.gizmag.com/scaneagle-uav-gets-synthetic-aperture-radar-sar/9007/. Date of access: 8 Aug. 2017.

Hart, S.G. & Wempe, T.E. 1979. Cockpit Display of Traffic Information : Airline Pilots' Opinions About Content, Symbology, and Format. *NASA Technical memorandum.*

Herwitz, S.R., Johnson, L.F., Dunagan, S.E., Higgins, R.G., Sullivan, D. V., Zheng, J., Lobitz, B.M., Leung, J.G., Gallmeyer, B.A., Aoyagi, M., Slye, R.E., & Brass, J.A. 2004. Imaging from an unmanned aerial vehicle: Agricultural surveillance and decision support. *Computers and Electronics in Agriculture*. 44(1):49-61.

Hjorland, B. 2009. Epistemology and Philosophy of Science for Information Scientist. URL http://www.iva.dk/jni/lifeboat/info.asp?subjectid=44. Date of access: 8 Aug. 2016.

Holness, C., Matthews, T., Satchell, K., & Swindell, E.C. 2016. Remote sensing archaeological sites through Unmanned Aerial Vehicle (UAV) imaging*. International Geoscience and Remote Sensing Symposium.* IEEE. pp. 6695-6698.

How, J., Frazzoli, E., & Chowdhary, G. 2015. Linear Flight Control Techniques for Unmanned Aerial Vehicles. 1st ed. *In*: Handbook of Unmanned Aerial Vehicle. Springer: pp. 529-576.

Huang, Y., Hoffmann, W.C., Lan, Y., Wu, W., & Fritz, B.K. 2009. Development of a spray system for an unmanned aerial vehicle platform. *Applied Engineering in Agriculture*. 25(6):803-810.

Huerta, M., 2013. Integration of civil unmanned aircraft systems (UAS) in the national airspace system (NAS) roadmap. *Federal Aviation Administration*. pp. 4-34.

Hunt, E.R., Walthall, C.L., Daughtry, C.S.T., Cavigelli, M.A., Fujikawa, S.J., Yoel, D.W., Ng, T.L., & Tranchitella, M.C. 2006. High-resolution multispectral digital photography using Unmanned Airborne Vehicles. *In*: Biennial Workshop on Aerial Photography,

Videography, and High Resolution Digital Imagery for Resource Assessment Proceedings. pp. 454-458.

Johannesson, P. & Perjons, E. 2014. An introduction to design science. Springer International Publishing.

Joint Capability Group on Unmanned Aerial Vehicles. 2010. Joint Doctrine Note 3 / 10 Unmanned Aircraft Systems : Terminology, Definitions and Classification.

Katsigiannis, P., Misopolinos, L., Liakopoulos, V., Alexandridis, T.K., & Zalidis, G. 2016. An autonomous multi-sensor UAV system for reduced-input precision agriculture applications. *In*: Mediterranean Conference on Control and Automation (MED), Athens, Greece: IEEE. pp. 60-64.

Kerr, S.C. 1998. Human induced loading on staircases. (Doctoral dissertation, University of London).

Kim, K., Jeon, M., Lee, J., Jeong, J., & Jeong, G. 2014. A study on the app development using sensor signals from smartphone and smart watch. Advanced Science and Technology Letters. 62:66-69.

Klimkowska, A., Lee, I., & Choi, K. 2016. Possibilities of uas for maritime monitoring. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences.* 41:885.

Komninos, A. & Dunlop, M. 2014. Text input on a smart watch. *IEEE Pervasive Computing.* 13(4):50-58.

Landis, J.R. & Koch, G.G. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics. JSTOR*. 33(1):159.

Larose, G. & Kruse, P. 2005. What is ontology? URL https://whatis.techtarget.com/definition/ontology. Date of access: 3 Aug. 2016.

Leylek, E.A. & Costello, M.F. 2012. Flight dynamic simulation for multibody aircraft configurations. *Journal of Guidance, Control, and Dynamics.* 35(6):1828-1842.

Lee, T., Leok, M., Mcclamroch, N.H., & Mar, O.C. 2010. Geometric Tracking Control of a Quadrotor UAV on SE (3). *In*: Conference on Decision and Control (CDC), Atlanta, GA, USA: IEEE. pp. 5420-5425.

Leishman, G.J., 2006. Principles of helicopter aerodynamics. Cambridge university press.

Lewis, M. 1998. Designing for Human-Agent Interaction. *AI Magazine*. 19(2):67.

Lindstr, V., Frohm, J., Winroth, M., & Stahre, J. 2008. Levels of automation in manufacturing. *Ergonomia*. 1:1-28.

Liu, Y., Li, Z., Hayward, R., Walker, R., & Jin, H. 2009. Classification of Airborne LIDAR Intensity Data Using Statistical Analysis and Hough Transform with Application to Power Line Corridors. *Digital Image Computing: Techniques and Applications*. 1:462-467.

Liu, M.-K., Lin, Y.-T., Qiu, Z.-W., Kuo, C.-K., & Wu, C.-K. 2020. Hand Gesture Recognition by a MMG-based Wearable Device. *IEEE Sensors Journal*. 1748(c):1-10.

Lu, Y., Han, F., Xie, L., Yin, Y., Shi, C., & Lu, S. 2017. I Am the UAV: A Wearable Approach for Manipulation of Unmanned Aerial Vehicle. *In*: International Conference on Smart Computing (SMARTCOMP), Hong Kong, China: IEEE: pp. 11-13.

Lucas, B.D. & Kanade, T. 1981. An iterative image registration technique with an application to stereo vision. *IJCAI*: pp. 674-679.

Ma'Sum, M.A., Arrofi, M.K., Jati, G., Arifin, F., Kurniawan, M.N., Mursanto, P., & Jatmiko, W. 2013. Simulation of intelligent Unmanned Aerial Vehicle (UAV) for military surveillance. *In*: International Conference on Advanced Computer Science and Information Systems (CACSIS), Bali, Indonesia: IEEE. pp. 161-166.

March, S.T. & Smith, G.F. 1995. Design and natural science research on information technology. *Decision Support Systems*. 15(4):251-266.

Marshall, D. 2009. Unmanned aerial systems and international civil aviation organization regulations. NDL Rev: (85):693.

Martin, G. 2012. Modelling and Control of the Parrot AR. Drone. *The UNSW Canberra at ADFA Journal of Undergraduate Engineering Research.* 5(1):12.

Mei, C., Sibley, G., Cummins, M., Newman, P., & Reid, I. 2011. RSLAM: A system for large-scale mapping in constant-time using stereo. *International Journal of Computer Vision.* 94(2):198-214.

Mejias, L., Lai, J., & Bruggemann, T. 2015. Sensors for Missions. *In*: Handbook of Unmanned Aerial Vehicles. Springer: pp. 385-399.

Merz, T., Duranti, S., & Conte, G. 2006. Autonomous landing of an unmanned helicopter based on vision and inertial sensing. *In*: Experimental Robotics IX. Springer: pp. 343-352.

Miller, I. & Campbell, M. 2008. Particle filtering for map-aided localization in sparse GPS environments. *In*: International Conference on Robotics and Automation, Pasadena, CA, USA: IEEE. pp. 1834-1841.

Mills, F. & Stufflebeam, R. 2005. Introduction to Intelligent Agents. Consortium on Cognitive Science Instruction. URL http://www.mind.ilstu.edu/curriculum/ants_nasa/intelligent_agents.php. Date of access: 23 Oct. 2017.

Mliki, H., Bouhlel, F., & Hammami, M. 2020. Human activity recognition from UAV-captured video sequences. Pattern Recognition. *Elsevier Ltd*. 100:107140.

Morier, B.Y. 2010. European Aviation Safety Agency. *In*: European Aviation Safety Agency (EASA) Handbook. pp. 2009-2011.

Mueller, T. 2001. Fixed and Flapping Wing Aerodynamics for Micro Air Vehicle Applications. *American Institute of Aeronautics and Astronautics*.

Mugnier, M. 2015. No piloting whatsoever. Hexo plus. URL https://hexoplus.com/no-piloting. Date of access: 15 Mar. 2016.

Nag, S., Jung, J., & Inamdar, K. 2017. Communicating with unmanned aerial swarm automatic dependent surveillance transponders. *IEEE*. 1:1-3.

Natarajan, K., Nguyen, T.H.D., & Mete, M. 2018. Hand gesture controlled drones: An open source library. *In*: International Conference on Data Intelligence and Security (ICDIS), South Padre Island, TX, USA: IEEE. pp. 168-175.

Needham, J. 1965. Science and Civilisation in China. 4th ed. Cambridge University Press.

Nigel. 2017. What is Pitch, Roll and Yaw? URL https://emissarydrones.com/what-is-roll-pitch-and-yaw. Date of access: 28 Sep. 2017.

Nunamaker, J., Chen, M., & Purdin, T. 1990. Systems development in Information Systems research. *Journal of Management Information Systems*. 7(3):89-106.

Oates, B.J. 2015. Researching information systems and computing. Sage.

Oughlin, N. 2012. The Benefits and Disadvantages of positivism in International Theory. URL http://www.e-ir.info/2012/01/20/what-are-the-benefits-and-disadvantages-of-positivism-for-international-theory/. Date of access: 5 Oct. 2016.

Padhy, R.P., Verma, S., Ahmad, S., Choudhury, S.K. and Sa, P.K., 2018. Deep neural network for autonomous uav navigation in indoor corridor environments. *Procedia computer science*. pp.643-650.

Papapetrou, P., Athitsos, V., Potamias, M., Kollios, G., & Gunopulos, D. 2011. Embedding-based subsequence matching in time-series databases. *ACM Transactions on Database Systems.* 36(3):1-39.

Parasuraman, R., Sheridan, T.B., & Wickens, C.D. 2000. A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*. 30(3):286-297.

Parrot. 2012. Parrot AR.Drone API. URL https://projects.ardrone.org/projects/show/ardrone-api. Date of access: 6 Jun. 2017.

Parrot. 2014. Parrot Bebop Drone. Lightweight yet robust quadricopter - 14 megapixel Full HD 1080p Fisheye Camera - Skycontroller - 3-axes image stabilization URL http://global.parrot.com/au/products/bebop-drone/. Date of access: 27 Sep. 2017.

Parrot. 2018. FreeFlight Pro - Android Apps on Google Play. URL https://play.google.com/store/apps/details?id=com.parrot.freeflight3&hl=en. Date of access: 27 Sep. 2017.

Pazouki, M., Wu, Z., Yang, Z., & Möller, D.P.F. 2015. An efficient learning method for RBF Neural Networks. *In*: International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland: IEEE. pp. 1-6.

Perry, J., Mohamed, A., Abd-Elrahman, A., Bowman, S., Kaddoura, Y., & Watts, A. 2008. Precision directly georeferenced unmanned aerial remote sensing system: Performance evaluation. *In*: Proceedings of the Institute of Navigation, National Technical Meeting. pp. 680-688.

Pippin, C. 2015. Integrated hardware/software architectures to enable uavs for autonomous flight. *In*: Handbook of Unmanned Aerial Vehicles. Springer: pp. 1725-1747.

Piskorski, S., Brulez, N., Drone, A., Guide, D., Eline, P., & D 'haeyer, F. 2012. Developer Guide SDK 2.0. *AR.Drone developer guide.*

Popescu, D., Ichim, L., & Caramihale, T. 2015. Flood areas detection based on UAV surveillance system. *In*: International Conference on System Theory, Control and Computing (ICSTCC), Cheile Gradistei, Romania: IEEE. pp. 753-758.

Porikli, F. & Yilmaz, A. 2012. Object Detection & Tracking. *In*: Video Analytics for Business Intelligence. Springer: pp. 3-41.

Priya, R. 2015. Comte's Positivism and Its Characteristics. URL http://www.yourarticlelibrary.com/sociology/comtes-positivism-and-its-characteristics/43730/. Date of access: 3 Aug. 2016.

Rafi, F., Khan, S., Shafiq, K., & Shah, M. 2006. Autonomous target following by unmanned aerial vehicles. *Defense and Security Symposium*. 6230:623010.

Ranasinghe, R.A.T.M., Jaksa, M.B., Kuo, Y.L., & Pooya Nejad, F. 2017. Application of artificial neural networks for predicting the impact of rolling dynamic compaction using dynamic cone penetrometer test results. *Journal of Rock Mechanics and Geotechnical Engineering. Elsevier Ltd*. 9(2):340-349.

Rao, K.R., Kim, D.N., & Hwang, J.J. 2011. Fast Fourier Transform: Algorithms and Applications. Springer.

Rawassizadeh, R., Price, B.A., & Petre, M. 2014. Wearables: Has the age of smartwatches finally arrived? *Communications of the ACM*. 58(1):45-47.

Reagan, J. 2014. Drone Definitions: Learning the Lingo of UAS. URL https://dronelife.com/2014/09/29/drone-definitions-learning-uas/. Date of access: 27 Sep. 2017.

Rising, L. & Janoff, N.S. 2000. The Scrum software development process for small teams. *IEEE software*. 17(4):26-32.

Roberts, Stirling, Zufferey, & Floreano. 2007. Quadrotor Using Minimal Sensing For Autonomous Indoor Flight. *In*: European Micro Air Vehicle Conference and Flight Competition (EMAV). pp. 1-8.

Rodkin, D. 2014. Center of gravity NASA. URL https://www.grc.nasa.gov/www/k-12/airplane/cg.html. Date of access: 27 Sep. 2017.

Rodríguez-Canosa, G.R., Thomas, S., del Cerro, J., Barrientos, A., & MacDonald, B. 2012. A Real-Time Method to Detect and Track Moving Objects (DATMO) from Unmanned Aerial Vehicles (UAVs) Using a Single Camera. *Remote Sensing*. 4(4):1090-1111.

Ross, S., Melik-Barkhudarov, N., Shankar, K.S., Wendel, A., Dey, D., Bagnell, J.A., & Hebert, M. 2013. Learning monocular reactive UAV control in cluttered natural environments. *In*: International Conference on Robotics and Automation, Karlsruhe, Germany: IEEE. pp. 1765-1772.

Rouse, M. 2015. Internet of things agenda. URL http://internetofthingsagenda.techtarget.com/definition/drone. Date of access: 15 Mar. 2016.

Russell, S. & Norvig, P. 2010. Artificial intelligence: a modern approach. Pearson Education.

Schwaber, K. 1995. Scrum development process. *In*: Business Object Design and Implementation. Springer: London, pp. 117-134.

Schwarz, B. 2010. Mapping the world in 3D. *Nature Photonics.* 4(7):429-430.

Sebbane, Y.B. 2015. Smart autonomous aircraft: Flight control and planning for UAV. *CRC Press*.

Shyy, W., Aono, H., Chimakurthi, S.K., Trizila, P., Kang, C.K., Cesnik, C.E.S., & Liu, H. 2010. Recent progress in flapping wing aerodynamics and aeroelasticity. *Progress in Aerospace Sciences*. 46(7):284-327.

Smith, S.W. 1997. The Fast Fourier Transform. *In*: The Scientist and Engineer's Guide to Digital Signal Processing. California Technical Publishing: pp. 225-232.

Stansbury, R.S., Vyas, M.A., & Wilson, T.A. 2009. A survey of UAS technologies for command, control, and communication (C3). *In*: Unmanned Aircraft Systems. Springer, pp. 61-78.

Stark, B. & Chen, Y. 2016. A framework of optimal remote sensing using small unmanned aircraft systems. *In*: International Conference on Mechatronic and Embedded Systems and Applications (MESA 2016), Auckland, New Zealand: IEEE. pp. 1-6.

Stevenson, J.D., O'Young, S., & Rolland, L. 2015. Assessment of alternative manual control methods for small unmanned aerial vehicles. *Journal of Unmanned Vehicle Systems. Elsevier B.V*. 3(3):73-94.

Stimson, G.W. 1998. Stimson's Introduction to Airborne Radar. *In*: Introduction to Airborne Radar. SciTech, p. 1381.

Susman, G. & Evered, R.D. 1978. An assessment of the scientific merits of action research. *Administrative science quarterly*. 23(4):582-603.

Suzuki, K. 2011. Artificial Neural Networks: Industrial and Control Engineering Applications. InTech: pp. 1-478

Tadema, J., Theunissen, E., & Koeners, J. 2007. Using perspective guidance overlay to improve UAV manual control performance. *In*: Enhanced and Synthetic Vision. p. 65590C.

Tahar, K.N. & Kamarudin, S.S. 2016. UAV on-board GPS in positioning determination. In: International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences. pp. 41.

Tahir, A., Böling, J., Haghbayan, M.-H., Toivonen, H.T., & Plosila, J. 2019. Swarms of Unmanned Aerial Vehicles - A Survey. *Journal of Industrial Information Integration. Elsevier Inc*. 16:100106.

Tao, K.M. 1993. A closer look at the radial basis function (RBF) networks. *In*: Proceedings of 27th Asilomar Conference on Signals, Systems and Computers. pp. 401-405.

Techopedia. 2019. What is six degrees of separation? - Definition from WhatIs.com. URL http://whatis.techtarget.com/definition/six-degrees-of-separation. Date of access: 25 Jun. 2017.

Thomasnet. 2018. Types of Actuators and Their Applications and Uses URL https://www.thomasnet.com/articles/pumps-valves-accessories/types-of-actuators/. Date of access: 26 Jan. 2018.

Tisdale, J., Ryan, A., Zennaro, M., Xiao, X., Caveney, D., Rathinam, S., Hedrick, J.K., & Sengupta, R. 2006. The software architecture of the berkeley UAV platform. *In*: International Conference on Control Applications, Munich, Germany: IEEE. pp. 1420-1425.

Toth, C. & Jóźków, G. 2016. Remote sensing platforms and sensors: A survey*. ISPRS Journal of Photogrammetry and Remote Sensing*. 115:22-36.

Tripicchio, P., Satler, M., Dabisias, G., Ruffaldi, E., & Avizzano, C.A. 2015. Towards Smart Farming and Sustainable Agriculture with Drones. *In*: International Conference on Intelligent Environments, Prague, Czech Republic: IEEE. pp. 140-143.

Trofin, C. 2015. Motion picture stabilizing achieved by mechanical engineering: shooting video using three axis camera gimbals. *Academic Journal of Manufacturing Engineering*. 13(3):70-81.

Trueman, C. 2015. *Positivism*. URL http://www.historylearningsite.co.uk/sociology/research-methods-in-sociology/positivism/. Date of access: 8 Oct. 2016.

Truncellito, D. 2015. Epistemology | Internet Encyclopedia of Philosophy. URL https://iep.utm.edu/epistemo/. Date of access: 3 Aug. 2016.

U.S. Department of Defense. 2000. Standard practice for system safety URL https://www.dau.edu/cop/armyesoh/DAU Sponsored Documents/MIL-STD-882E.pdf. Date of access: 27 Jan. 2017.

Vaishnavi, V. & Kuechler, B. 2004. Design Science Research in Information Systems. In Design research in information systems. 1:45.

Valavanis, K.P., Astuti, G., Caltabiano, D., Giudice, G., Longo, D., Melita, D., Muscato, G., & Orlando, A. 2007. Advances in Unmanned Aerial Vehicles. *In*: Symposium on Intelligent Unmanned System October. Springer Publishing Company: Netherlands, pp. 15-46.

Valavanis, K.P. & Vachtsevanos, G.J. 2014. Military and Civilian Unmanned Aircraft. *In*: Handbook of Unmanned Aerial Vehicles. Springer: pp. 94-103.

Valavanis, K.P. & Vachtsevanos, G.J. 2015. Handbook of unmanned aerial vehicles. Springer: pp:1-3022.

Ventura, D., Bruno, M., Jona Lasinio, G., Belluscio, A., & Ardizzone, G. 2016. A low-cost drone based application for identifying and mapping of coastal fish nursery grounds. *Estuarine, Coastal and Shelf Science. Elsevier Ltd*. 171:85-98.

Vergara, J. 2015. Moto 360 (2nd Gen.) review. Android Authority. URL http://www.androidauthority.com/moto-360-2nd-gen-review-648624/. Date of access: 6 Jun. 2017.

Vincenzi, D.A., Terwilliger, B.A., & Ison, D.C. 2015. Unmanned Aerial System (UAS) Human-machine Interfaces: New Paradigms in Command and Control. *Procedia Manufacturing. Elsevier B.V*. 3:920-927.

Vu, A., Member, S., Ramanandan, A., Chen, A., Farrell, J.A., Barth, M., & Member, S. 2012. Real-Time Computer Vision / DGPS-Aided Inertial Navigation System for Lane-Level Vehicle Navigation. *IEEE Transactions on Intelligent Transportation Systems*. 13(2):899-913.

Waharte, S. & Trigoni, N. 2010. Supporting search and rescue operations with UAVs. *In*: International Conference on Emerging Security Technologies, Canterbury, UK: IEEE. pp. 142-147.

Waley, A. 1936. Suggestion concerning the origin of the kite, in reviewing Bodde. *In*: Science and Civilisation in China. Cambridge University Press.

Walker, M., 1975. Hot-Air Balloons. *Science and Children*, 12(5), pp.18-19.

Watt, K. 2010. Positivism and the "Enlightenment". URL http://scienceandsystems.co.za/2010/03/positivism-and-enlightenment.html. Date of access: 9 Oct. 2016.

Weiss, G.M., Timko, J.L., Gallagher, C.M., Yoneda, K., & Schreiber, A.J. 2016. Smartwatch-based activity recognition: A machine learning approach. *In*: International Conference on Biomedical and Health Informatics (BHI), Las Vegas, NV, USA: IEEE. pp. 426-429.

Wickens, C.D. & Hollands, J.G. 2000. Automation and human performance. *In*: Engineering Psychology and Human Performance. Pearson Education, p. 544.

Williams, L. 2002. Agile Software Development. *Computer Science Education*. 12(3):167-168.

Witten, I.H., Frank, E., & Hall, M. a. 2011. Data Mining: Practical Machine Learning Tools and Techniques. 2nd ed. *In*: Complementary literature None. Elsevier.

Xu, C., Pathak, P.H., & Mohapatra, P. 2015. Finger-writing with Smartwatch: A Case for Finger and Hand. In: International Workshop on Mobile Computing Systems and Applications. pp. 9-14.

Yu, H., Xie, T., Paszczyñski, S., & Wilamowski, B.M. 2011. Advantages of radial basis function networks for dynamic system design. *IEEE Transactions on Industrial Electronics.* 58(12):5438-5450.

Zaheer, Z., Usmani, A., Khan, E., & Qadeer, M.A. 2016. Aerial surveillance system using UAV. *In*: 2016 International Conference on Wireless and Optical Communications Networks (WOCN), Hyderabad, India: IEEE. pp. 1-7.

Zemouri, R., Racoceanu, D., & Zerhouni, N. 2003. Recurrent radial basis function network for time-series prediction. *Engineering Applications of Artificial Intelligence*. 16(5-6):453-463.

Zhang, G. & Hsu, L.T. 018. Intelligent GNSS/INS integrated navigation system for a commercial UAV flight control system. *Aerospace Science and Technology. Elsevier Masson SAS*. 80:368-380.

ANNEXURE A – JAVA CODE SAMPLES

```
/*
 * Reads the x, y, z axis accelerometer data of the smartwatch
 * and send the data to the smartphone
 */

public void onSensorChanged(SensorEvent sensorEvent) {
    float[] data = sensorEvent.values;
    putDataMapReq = PutDataMapRequest.create("/SAMPLE");
    putDataMapReq.getDataMap().putFloatArray(KEY,data);
    putDataReq = putDataMapReq.asPutDataRequest();
    resultado = Wearable.DataApi.putDataItem(mApiClient,putDataReq);
    Wearable.DataApi.putDataItem(mApiClient,putDataReq);
}

/*
 * Gets the x, y, z axis accelerometer data from the smartwatch.
 * Apply FFT analysis on the data.
 * Perform RBFNN prediction.
 */

public void onDataChanged(final DataEventBuffer dataEventBuffer) {
    //listens for data changes from smartwatch
    for (DataEvent event : dataEventBuffer {
     if ((event.getType() == DataEvent.TYPE_CHANGED) &&(test)) {
        DataItem item = event.getDataItem();
        if (item.getUri().getPath().equals("/SAMPLE")) {
        DataMapItem dataMapItem = DataMapItem.fromDataItem(item);
        acel_xyz = dataMapItem.getDataMap().getFloatArray(KEY);
        runOnUiThread(new Runnable() {
        @Override
        public void run() {
        //store the accelerometer data received from the smartwatch
        xData.add(acel_xyz[0]);
        yData.add(acel_xyz[1]);
        zData.add(acel_xyz[2]);
        dataSize = xData.size();
          timer.postDelayed(new Runnable() {
             @Override
              public void run() {
              //check if prediction should be performed
              if(predict) {
              //create imaginary values for FFT analysis
              imaginaryX = new double[dataSize];
              imaginaryY = new double[dataSize];
              imaginaryZ = new double[dataSize];
              //perform FFT analysis on the accelerometer data
              double[] fftValueX = performFFT(fft, xData, imaginaryX);
              double[] fftValueY = performFFT(fft, yData, imaginaryY);
              double[] fftValueZ = performFFT(fft, zData, imaginaryZ);
              //gather all FFT values
              double[] fftArray = new double[9];
              System.arraycopy(fftValueX, 0, fftArray, 0, 3);
              System.arraycopy(fftValueY, 0, fftArray, 3, 3);
              System.arraycopy(fftValueZ, 0, fftArray, 6, 3);
              //perform prediction
              pilotWithPrediction(neuralNetworkPrediction(fftArray));
```

```
}
   else {
        //clear data
        xData = new ArrayList<>();
        yData = new ArrayList<>();
        zData = new ArrayList<>();
        imaginaryX = new double[0];
        imaginaryY = new double[0];
        imaginaryZ = new double[0];
        timer.removeCallbacksAndMessages(null);
        }
           }
            //timer is set for 5 seconds
              }, 5000);
                }
                 });
                }
           }
       }
}

/*
 * Apply FFT analysis on the data.
 */

public static double[] performFFT(Fft fft, double[] re, double[] im) {
    fft.transform(re,im);
    double[] fftData = new double[3];
    for (int i = 0; i < 3; i++) {
        fftData[i]= (int) (Math.sqrt(im[i]*im[i] + re[i]*re[i])  * 1000) / 1000.0;
    }
    return fftData;
}

/*
 * Neural network prediction based on FFT data received.
 */

public String neuralNetworkPrediction(double[] fftArray){
    //setup data for neural network
    RBFData[] rbfData = new RBFData[]{
            new RBFData(fftArray)
    };
    StringBuilder sb = new StringBuilder("Test Data:\n");
    for(RBFData s : rbfData) {
        sb.append(s.toString() + "\n");
    }
    //get trained model from rescources
    try {
        mClassifier = (Classifier)
weka.core.SerializationHelper.read(assetManager.open("RBFModel.model"));
    } catch (IOException e) {
        e.printStackTrace();
    } catch (Exception e) {
        e.printStackTrace();
    }
    //set attributes for neural network
    final Attribute X1 = new Attribute("X1"); // x - axis
    final Attribute X2 = new Attribute("X2");
    final Attribute X3 = new Attribute("X3");
    final Attribute Y1 = new Attribute("Y1"); // y - axis
```

```
    final Attribute Y2 = new Attribute("Y2");
    final Attribute Y3 = new Attribute("Y3");
    final Attribute Z1 = new Attribute("Z1"); // z - axis
    final Attribute Z2 = new Attribute("Z2");
    final Attribute Z3 = new Attribute("Z3");
    //set classes for prediction
    final List<String> classes = new ArrayList<String>() {
        {
            add("Circle");
            add("Up Down");
            add("Left Right");
        }
    };
    //perform prediction
    try {
        double result = mClassifier.classifyInstance(newInstance);
        String className = classes.get(new Double(result).intValue());
        String prediction = "Predicted: " + className;
        return prediction;
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}

/*
 * Pilot the UAS based on the prediction made.
 *
 */

public void PilotWithPrediction(String prediction) {
    if (prediction.equals("Predicted: Circle")) {
        //get the flying state in order to ensure the control task can be performed
        switch (mBebopDrone.getFlyingState()) {
        case ARCOMMANDS_ARDRONE3_PILOTINGSTATE_FLYINGSTATECHANGED_STATE_LANDED:
         //send take off notification to smartwatch
         sendAsyncMessage(WEAR_MESSAGE_PATH, "Take off");
         mBebopDrone.takeOff();
         break;
        case ARCOMMANDS_ARDRONE3_PILOTINGSTATE_FLYINGSTATECHANGED_STATE_HOVERING:
         //send landed notification to smartwatch
         sendAsyncMessage(WEAR_MESSAGE_PATH, "Landed");
         mBebopDrone.land();
         break;
        default:
        }
    }
    //follow the pilot based on gps coordinates
    else if(prediction.equals("Predicted: Up Down")){
        switch (mBebopDrone.getFlyingState()) {
        case ARCOMMANDS_ARDRONE3_PILOTINGSTATE_FLYINGSTATECHANGED_STATE_LANDED:
        //send take off notification to smartwatch
         sendAsyncMessage(WEAR_MESSAGE_PATH, "Take off");
         mBebopDrone.takeOff();
        case ARCOMMANDS_ARDRONE3_PILOTINGSTATE_FLYINGSTATECHANGED_STATE_FLYING:
         //send following notification to smartwatch
         sendAsyncMessage(WEAR_MESSAGE_PATH, "Following");
         mBebopDrone.PilotWithGPS(latitude, longitude, altitude);
```

```
        case ARCOMMANDS_ARDRONE3_PILOTINGSTATE_FLYINGSTATECHANGED_STATE_HOVERING:
         //send following notification to smartwatch
         sendAsyncMessage(WEAR_MESSAGE_PATH, "Following");
         mBebopDrone.PilotWithGPS(latitude, longitude, altitude);
      break;
     default:
      }
   }
}

/*
 * Get the pilots location data.
 */

public void onLocationChanged(Location location) {
    latPilot = location.getLatitude();
    longPilot = location.getLongitude();
    altPilot = location.getAltitude();
}

/*
 * Navigate the UAS to the pilots location.
 */

public void PilotWithGPS(double latPilot, double longPilot, double altPilot) {
  //check if distance between pilot and UAS is sufficient
  boolean sufficientDistance = calculateDistance(latPilot, longPilot, latUAS,
  longUAS);
    if(mDeviceController != null) && (sufficientDistance) {
        mDeviceController.getFeatureARDrone3().
         //UAS navigates to GPS coordinates
        sendPilotingMoveTo((double)latPilot, (double)longPilot, (double)altPilot,
        (ARCOMMANDS_ARDRONE3_PILOTING_MOVETO_ORIENTATION_MODE_ENUM.
        ARCOMMANDS_ARDRONE3_PILOTING_MOVETO_ORIENTATION_MODE_TO_TARGET),0);
    }
}

/*
 * Calculates the distance between the UAS and the pilot to determine if the
 * distance is sufficient.
 */

public static boolean calculateDistance(double latPilot, double longPilot,
                                        double latUAS, double longUAS) {

    Location locationPilot = new Location("point A");
    locationPilot.setLatitude(latPilot);
    locationPilot.setLongitude(longPilot);
    Location locationUAS = new Location("point B");
    locationUAS.setLatitude(latUAS);
    locationUAS.setLongitude(longUAS);
    //calculates the distance between the pilot and the UAS
    float distance = locationPilot.distanceTo(locationUAS);
    //if the distance is equal or more than 5 meters return true
    if (distance >= 5) {
        return true;
    } else {
        return  false;
    }
}
```

```
/*
 * Adjust the speed of the UAS based on the activity prediction.
 */

public void PilotWithSpeed(String prediction) {
    switch (prediction) {
        case "standing":
            //adjust speed by setting the pitch value.
            mBebopDrone.setPitch((byte) 0);
        case "walking":
            mBebopDrone.setPitch((byte) 50);
        case "running":
            mBebopDrone.setPitch((byte) 100);
        default:
            mBebopDrone.emergency();
    }
}
```

ANNEXURE B – RESULTS COMPARISON

										Prediction	
Gesture	X-1	X-2	X-3	Y-1	Y-2	Y-3	Z-1	Z-2	Z-3	PC	Phone
1	472.89	790.59	225.79	561.19	11.62	211.55	715.35	238.31	245.59	1	1
1	506.59	623.36	236.15	461.44	59.92	139.84	601.36	182.96	188.44	1	1
1	644.54	666.96	284.99	423.35	59.95	135.99	614.46	271.95	179.07	1	1
1	438.94	688.45	203.20	420.11	27.42	141.33	681.52	169.89	259.56	1	1
1	466.60	665.68	198.87	538.22	6.07	129.17	708.46	175.88	226.30	1	1
1	360.15	587.09	134.10	421.39	54.01	129.06	600.87	235.52	152.49	2	2
1	454.91	722.47	163.94	458.50	158.87	133.01	827.52	339.78	218.72	1	1
1	379.74	638.66	134.96	547.87	27.25	148.95	707.66	236.07	202.67	1	1
1	604.41	691.51	243.52	566.19	120.99	187.03	785.55	311.37	217.69	1	1
1	608.10	729.72	208.89	590.54	96.66	111.11	895.00	371.31	211.41	1	1
2	481.83	449.07	197.96	353.97	65.08	88.19	467.87	199.53	106.61	2	2
2	420.96	410.94	189.35	294.16	65.82	58.39	359.73	161.84	87.76	2	2
2	433.75	430.42	191.93	360.91	46.75	79.38	405.97	170.11	132.71	2	2
2	437.08	393.48	182.73	252.47	38.85	98.09	384.03	169.26	98.16	2	2
2	483.71	413.22	190.50	351.31	37.74	115.84	385.79	188.75	85.12	2	2
2	365.24	375.01	158.75	349.84	83.15	100.74	486.78	126.28	68.43	2	2
2	397.64	398.60	179.98	429.52	70.11	114.46	447.65	155.08	59.11	2	2
2	352.78	163.85	126.57	778.40	42.81	49.42	680.70	157.57	91.63	2	2
2	391.21	428.21	119.31	442.45	55.74	117.09	659.53	182.22	66.17	2	2
2	465.03	585.18	158.16	534.52	16.60	152.80	691.53	221.52	93.86	2	2
3	307.42	68.83	44.79	127.69	35.73	32.52	232.48	71.11	45.01	3	3
3	446.21	111.43	41.67	168.95	82.83	112.82	392.30	100.20	14.05	3	3
3	244.40	115.09	88.48	109.51	149.34	84.35	393.62	88.07	50.57	3	3
3	239.77	125.32	61.62	130.14	94.27	58.90	376.42	66.17	46.88	3	3
3	189.81	84.91	63.64	93.93	107.25	69.09	319.05	66.74	34.37	3	3
3	222.59	97.75	73.91	51.20	109.50	60.59	351.72	74.35	36.81	3	3
3	266.08	149.37	82.36	126.44	117.31	104.85	441.09	94.69	42.60	3	3
3	160.50	132.98	48.38	128.33	81.40	30.01	313.20	63.49	30.77	3	3
3	202.61	143.45	46.75	165.44	94.72	37.37	362.28	85.75	45.35	3	3
3	390.07	200.81	65.01	196.52	100.77	76.75	506.94	109.80	4.13	3	3

Table A-1: Comparison between Weka results on the computer and mobile application for gesture recognition

											Prediction	
Activity	1	2	3	4	5	6	7	8	9	10	PC	Phone
1	3.41	1.73	10.17	5.24	2.89	2.75	2.55	1.58	1.60	1.48	1	1
1	5.07	6.49	3.99	1.43	0.95	1.90	1.34	0.42	0.54	0.66	1	1
1	7.59	6.17	2.27	2.11	1.24	0.56	0.27	1.08	0.83	0.83	1	1
1	1.54	0.74	2.13	2.88	1.52	1.07	0.64	0.25	0.45	0.20	1	1
1	4.46	6.94	4.77	1.97	2.24	0.93	0.48	0.46	0.69	0.83	1	1
1	5.29	2.95	2.51	1.15	0.69	1.48	0.78	0.52	0.60	0.59	1	1
1	3.08	0.57	0.39	0.51	0.30	0.16	0.29	0.38	0.37	0.12	1	1
1	0.91	0.85	1.05	0.69	0.54	0.42	0.20	0.59	0.10	0.55	1	1
1	1.85	0.15	1.24	1.32	0.96	0.34	0.04	0.57	0.12	0.23	1	1
1	3.99	0.99	1.09	1.14	0.69	0.71	0.40	0.24	0.49	0.35	1	1
2	14.85	32.63	7.91	9.47	5.91	4.00	2.35	2.74	2.24	1.74	2	2
2	3.46	9.57	11.30	8.82	6.24	0.82	1.44	0.71	0.49	0.81	2	2
2	19.85	10.56	2.90	4.21	9.62	7.10	5.04	4.38	3.16	1.93	2	2
2	49.99	25.90	24.54	6.94	13.18	8.29	3.38	2.72	0.25	3.27	2	2
2	22.32	28.20	29.47	29.58	17.67	18.23	11.25	10.60	9.53	7.34	2	2
2	41.08	33.61	37.68	28.31	27.19	20.68	16.28	10.88	6.40	6.10	2	2
2	42.17	20.95	24.56	14.06	2.79	4.39	3.97	1.18	0.97	1.47	2	2
2	64.79	26.36	4.96	10.16	2.06	0.97	1.29	2.64	1.49	1.97	2	2
2	125.07	31.05	17.89	1.35	6.69	1.99	2.67	2.52	0.77	1.54	3	3
2	14.61	7.97	16.54	12.25	3.67	11.37	6.92	4.43	4.11	2.19	2	2
3	27.05	12.34	12.72	18.56	15.82	7.87	4.48	3.62	3.59	1.47	3	3
3	14.43	18.18	9.31	21.32	14.61	7.53	2.26	2.30	1.72	3.52	2	2
3	146.71	52.88	28.10	50.92	19.75	24.52	28.18	12.91	12.95	10.49	3	3
3	79.52	45.09	31.28	35.11	49.08	53.92	17.54	14.12	4.71	6.52	3	3
3	54.90	7.20	22.73	19.82	22.41	7.08	4.80	2.67	5.88	6.07	3	3
3	62.19	17.06	48.50	31.55	17.35	7.89	22.40	15.56	3.68	5.84	3	3
3	39.62	40.19	30.77	17.81	26.34	23.22	6.11	2.66	3.90	1.83	3	3
3	39.21	8.92	33.49	45.52	28.23	19.55	14.67	13.13	0.31	6.10	3	3
3	36.24	81.43	59.56	38.45	48.95	64.00	42.96	11.19	9.43	8.48	3	3
3	149.32	66.79	106.97	44.94	23.27	26.70	25.81	27.01	17.83	17.01	3	3

Table A-2: Comparison between Weka results on the computer and mobile application for activity recognition

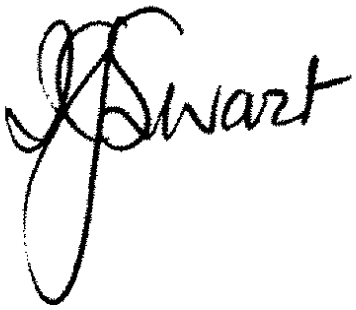
ANNEXURE C – CONFIRMATION OF LANGUAGE EDITING

This serves to confirm that I, Isabella Johanna Swart, registered with and accredited as professional translator by the South African Translators' Institute, registration number 1001128, language edited the following dissertation (excluding the Bibliography):

Simplifying UAS control and improving automation using an RBFNN for gesture and activity recognition

by

Adriaan van Schalkwyk



Dr Isabel J Swart

Poinsettia Close
Van der Stel Park
Dormehlsdrift GEORGE
6529
Tel: (044) 873 0111
Cell: 082 718 4210
e-mail: isaswart@telkomsa.net