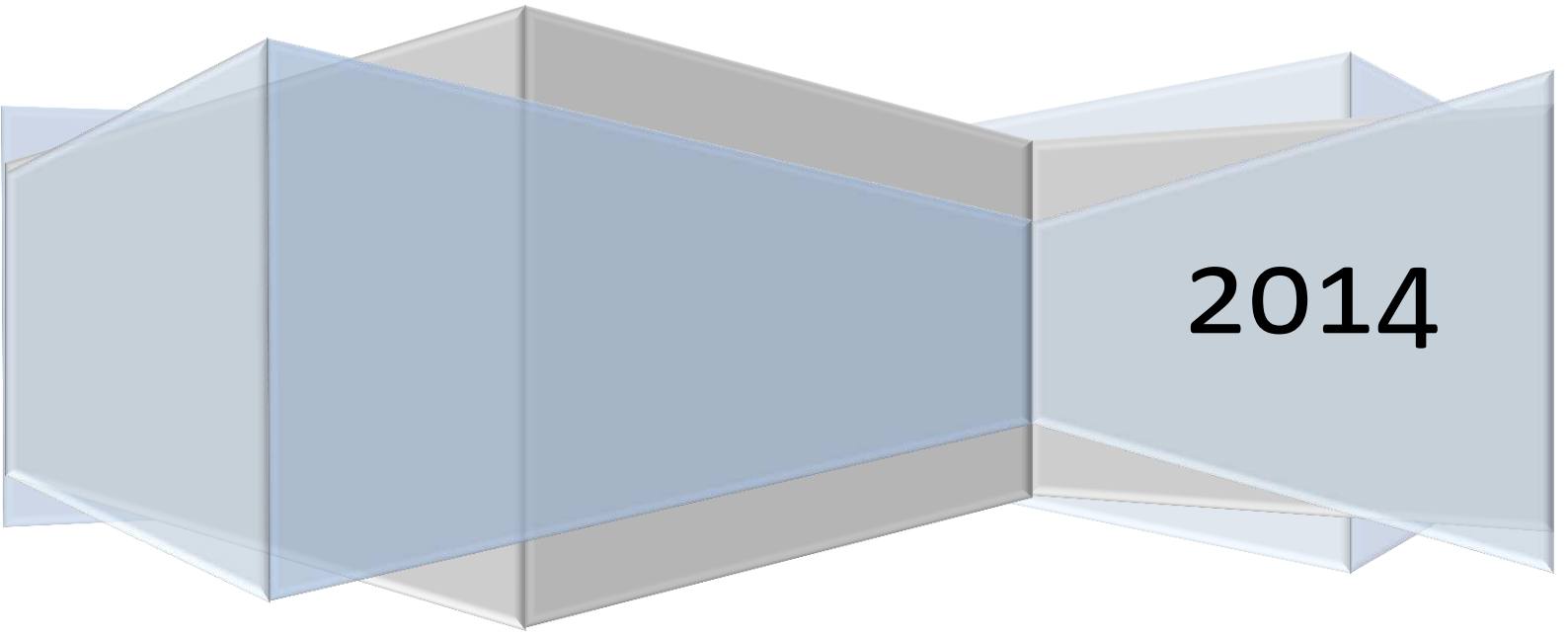
Autonomous Navigation and Search in an Indoor Environment Using an AR Drone

An Undergraduate Dissertation in Computer Science

By Erik Barrow



Abstract

In many situations the use of an robot such as an flying drone are used to help compliment humans in performing tasks such as search and rescue. Often these robots require a human operator to control and oversee them. The use of an autonomous robot that can navigate in unknown environments autonomously removes the need for a human operator and allows for quicker task performing. These autonomous drones need to be able to identify objects and navigate an unknown space without the interaction from a human operator. This paper presents visual processing algorithms that can be used with an AR Drone quad copter to identify objects and colour, the paper also presents some search patterns as possible methods for autonomously searching unknown spaces. At the end of the paper some experiments are presented with their results on possible solutions for autonomous search and navigation.

Acknowledgments

I would like to thank both my main supervisors for help they have provided with my project. Carey Pridgeon for helping me with choosing and selecting my dissertation subject, and refining the brief of what I would undertake. Also Amanda Brooks for her guidance in the layout and contents of my report, as well as constant feedback and support during the life span of my project.

Contents

Student Details i
Abstract ii
Attestation iii
Acknowledgments iv
Contents v
Figures list xi
1.0 Introduction 1
1.1 Aims and Objectives 2
1.2 Ethical and Legal Implications 2
1.2.1 Ethical Debate on Drones 2
1.2.2 Ethics and legalities of my project 3
1.3 Motivations 4
1.4 Work Breakdown Structure 5
1.5 Risk list 5
1.6 Scheduling 5
2.0 Literature Review 6
2.1 Introduction 6
2.2 Reviews 6
2.2.1 ArduCopter 6
2.2.2 Amazon Prime Air 6
2.2.3 Autonomous Navigation, object detection, and retrieval in an indoor environment 7
2.2.4 Framework for autonomous on-board navigation with the AR.Drone 8
2.2.5 Low cost MAV platform AR-drone in experimental verifications of methods for vision based autonomous navigation 9
2.2.6 Visual tracking and following of a quadrocopter by another quadrocopter 10
2.2.7 Autonomous indoor helicopter flight using a single on board camera 10
2.3 Conclusions and Findings 11
3.0 Research Methodologies 12

3.1 Software Planning 12
3.2 Algorithm Selection 12
3.2.1 Initial Selection Methods 13
3.2.1.1 Image Processing 13
3.2.1.2 Navigation 13
3.2.2 Algorithm Testing Methods 14
3.2.2.1 Image processing 14
3.2.2.2 Navigation 14
3.3 Evaluation 15
4.0 Research 16
4.1 Secondary Research 16
4.1.1 Algorithms 16
4.1.1.1 Image Processing 16
4.1.1.1.1 A look into image processing 16
4.1.1.1.2 Pre-Processing 16
4.1.1.1.3 The Algorithms 18
4.1.1.2 Autonomous Navigation 19
4.1.1.2.1 A look into Autonomous search 19
4.1.1.2.2 The Algorithms 20
4.1.2 Hardware Capabilities 25
4.1.2.1 Drone Capabilities 25
4.1.2.1.1 Camera 25
4.1.2.1.2 Sonar 26
4.1.2.1.3 Gyroscope 26
4.1.2.1.4 Compass 26
4.1.2.1.5 Wi-Fi 26
4.1.2.1.6 Weight Limit 26
4.1.2.2 Extra Hardware Available 26
4.1.2.2.1 Arduino 26
4.1.2.2.2 Raspberry Pi 26

4.1.2.2.3 Sensors 27
4.1.2.3 Sensor Configurations 29
4.1.3 Software 31
4.1.3.1 Programming languages used 31
4.1.3.2 Operating systems 31
4.1.3.3 APIs 31
4.1.3.3.1 AR Drone 2.0 API 32
4.1.3.3.2 Node JS Node-Copter 32
4.1.3.3.3 Open CV 32
4.1.3.3.4 Robot operating system 32
4.1.3.3.5 Drone API Choice 32
4.2 Primary Research 33
4.2.1 Testing Platform 33
4.2.2 Image Processing 33
4.2.2.1 Complications 33
4.2.2.2 Images 33
4.2.2.3 Template Matching 34
4.2.2.4 Pixel Classification 35
4.2.2.5 Results 36
4.2.2.6 Conclusion 37
4.2.3 Autonomous Navigation 40
4.2.3.1 Layouts 40
4.2.3.1.1 Layout A 40
4.2.3.1.2 Layout B 41
4.2.3.1.3 Layout C 41
4.2.3.2 Complications 42
4.2.3.3 Class Diagram 42
4.2.3.4 Virtual Layouts 43
4.2.3.5 Spiral 47
4.2.3.6 Wall Following 47

4.2.3.7 Zone 47
4.2.3.8 Results 48
4.2.3.9 Conclusions 48
5.0 Final Implementation 49
5.1 Prototype Design 49
5.1.1 Hardware 49
5.1.2 Design 49
5.1.2.1 Build 49
5.1.2.2 Wiring Schematic 50
5.1.3 Framework 51
5.2 Prototype Building 53
5.2.1 Building 53
5.2.2 Raspberry Pi Setup 54
5.3 Prototype Testing 55
5.3.1 Stable Flight (Pre-programming) 55
5.3.2 Sonar Sensors 57
5.3.3 Raspberry Pi Issues 58
6.0 Conclusion 59
6.1 Summary 59
6.2 Evaluation 59
6.2.1 Project 59
6.2.2 Project Management 59
6.2.2.1 Viva 60
6.3 Recommendations 60
6.4 Future Work 60
6.4.1 Heat camera 60
6.4.2 Larger drone 61
6.4.3 Face recognition 61
6.4.4 GPS 61
6.4.5 Speakers 61

6.4.6 Object Detection 61
6.4.7 Object Retrieval 61
6.4.8 Light Sensor 61
6.4.9 Advanced Search Patterns 62
6.4.10 Mobile App 62
7.0 Bibliography 63
8.0 Appendix 66
8.1 Meeting Records 66
8.1.1 Carey Pridgeon 66
8.1.2 Amanda Brooks 66
8.2 Code 67
8.2.1 Sonar Sensor Test 67
8.2.2 Image Processing 69
8.2.2.1 Template matching 69
8.2.2.2 Pixel Classification 71
8.2.3 Autonomous Navigation 72
8.3 Viva 82
8.3.1 Presentation 82
8.3.2 Feedback 91
8.3.2.1 Presentation 91
8.3.2.2 Project 91
8.4 Video Links 92
8.4.1 Concept Video 92
8.5 Word Count 92
8.6 Project Schedule 93
8.6.1 Expected 93
8.6.2 Actual 95
8.7 Work Breakdown Structure 97
8.8 Risk List 98
8.9 Project Proposal 99

Figures list

Figure 1 - The Air Navigation Order (2009) Article 166 3
Figure 2 - The Air Navigation Order (2009) Article 167 4
Figure 3 – Work Breakdown Structure 5
Figure 4 - Overall System Architecture [6] 8
Figure 5 - System Architecture [9] 9
Figure 6 – Hybrid methodology for my project 12
Figure 7 – Image algorithm comparison table 13
Figure 8 – Navigation Algorithm Comparison Table 14
Figure 9 – Image Test Table 14
Figure 10 – Navigation Test Table 15
Figure 11 – Noise Reduction [17] 16
Figure 12 – Sharpening [16] 17
Figure 13 - Constrained least squares filtering [16] 17
Figure 14 – Template Matching [18] 18
Figure 15 – Pixel classification [19] 19
Figure 16 – Spiral Search [20] 20
Figure 17 – Random Sampling [21] 21
Figure 18 - Grid Search [20] 22
Figure 19 – Zone [20] 22
Figure 20 – Parallel [20] 23
Figure 21 – Wall Following [22] 24
Figure 22 – Search Pattern comparison table 25
Figure 23 – Raspberry Pi Pin Diagram [25] 27
Figure 24 – Sonar Sensor Schematic [27] 28
Figure 25 – NoIR camera [28] 28
Figure 26 – IR distance Sensor [29] 29
Figure 27 – Sensor Configurations 31
Figure 28 33
Figure 29 33
Figure 30 33
Figure 31 33
Figure 32 33
Figure 33 33
Figure 34 33
Figure 35 33
Figure 36 33
Figure 37 33

Figure 38 33
Figure 39 33
Figure 40 34
Figure 41 34
Figure 42 34
Figure 43 34
Figure 44 34
Figure 45 34
Figure 46 34
Figure 47 34
Figure 48 – Table of Images 34
Figure 49 – Template Match Pass 34
Figure 50 – Template Match Fail 34
Figure 51 – Pixel Correlation Program 35
Figure 52 – Results Table 37
Figure 53 – Overall Pass Rate 37
Figure 54 – Distance Pass Rate 38
Figure 55 – Flight and Lighting Comparison 38
Figure 56 – Colour Key 40
Figure 57 – Layout A 40
Figure 58 – Layout B 41
Figure 59 – Layout C 41
Figure 60 – Class Diagram for search program 42
Figure 61 – Layout A 43
Figure 62 – Layout B 44
Figure 63 – Layout C 45
Figure 64 – Visibility Area 46
Figure 65 – Search Algorithm Results 48
Figure 66- Search pattern tests 48
Figure 67 – AR Drone Build Schematic [38] 49
Figure 68 – Wiring Diagram 50
Figure 69 – System Architecture 51
Figure 70 – Program Flow 52
Figure 71 – Drone Build and Circuitry 53
Figure 72 – Drone Build and Circuitry 54
Figure 73 – Remote connection 55
Figure 74 – Drone configuration file 56
Figure 75 – Reconfigured Shell 57
Figure 76 – Sonar Test 58

1.0 Introduction

The field of autonomous robots is still in its early years, with much advanced autonomous robotics being represented in science fiction, many of these science fictions are slowly becoming a reality. From current autonomous robots such as vacuum cleaners all the way up to UAV drones the field of research is vast with possibility. Currently there are minimal UAV drones that operate without the interaction of a human counterpart (such as those used by the Americans, have a pilot somewhere), autonomous drones that can operate without human input can be used in environments that are potentially dangerous to humans or in an everyday environment without the need to a pilot. Tasks for such an autonomous drone could be search and rescue, autonomous mapping, wildlife observation, aerial filming, testing of contaminated areas before human entry, and replacing humans in potentially dangerous situations. While there are few solutions out there, there are not many that both auto-navigate while also preforming a search. Searching also comes in many degrees of difficulty from finding block colours, to an object and face recognition.

This project proposes using an AR Drone (parrot) to produce a solution to an autonomous search robot. Equipped with extra sensors and using AI or agent based programming the drone will be required to roam an indoor environment searching for colours, objects, and people. The project will look into the scope of what objects can easily be detected using the on board camera, as well as other on board sensors, and which image processing algorithms are best for doing this both accurately and quickly on a moving platform. The project also covers looking into navigating an unknown indoor environment using search algorithms, testing and which of a variety of algorithms work best for a hovering UAV. Eventually the drone should be able to given an image of an object or person to find and perform autonomous actions to locate that object or person, without having an altercation with the environment around it.

A prime example of this in science fiction is that of the drones in the Terminator film series. While displayed in highly unlikely circumstances the variety of drones, including those that fly, are required to be able to autonomously navigate an environment and locate persons or objects. I plan to replicate this on a lower level, in a more ethical purpose and environment to that of science fiction films. This field of research is also the fuel for current debate on the ethics of how far an autonomous robot should be allowed to operate without human interaction or control.

1.1 Aims and Objectives

The overall aim of the project is to research and investigate into methods of autonomous search and navigation, determine the best solutions and to implement these solutions with the use of an AR Drone to a minimum point that the Drone can navigate itself and find a sheet of block colour.

- A look into the ethics of autonomous robotics, and current legislation and regulations that govern the use of autonomous robotics.
- Research into current autonomous search, navigation and object detection methods and algorithms.
- Identify Current case studies and research into autonomous flight
- Identify the best methods for autonomous tasks based on earlier research for a hovering quad copter.
- Design a prototype for the AR Drone and acquire the sensors and parts to build the prototype.
- Develop and code the prototype based of the algorithms researched
- Conduct field tests searching for an object without colliding with the environment
- Draw conclusions of the success/failure of the produced solution and make recommendations for future progression of the field.
- Critically evaluate the project

1.2 Ethical and Legal Implications

1.2.1 Ethical Debate on Drones

At the current time there is a variety of ethical implications that have been bought to the forefront of the news, this is mainly the privacy concerns related to the use of drones, but is not limited to drones, as articles such as Google glass have been reviewed over privacy concerns of cameras in public places.

A website online suggests some ethical codes over drone use in journalism [1]. The ethics code they put forwards suggests 5 levels of importance for ethical conduct, 3 of them relevant to my project (level 2-4). The second level, being the most important for me is “safety”. Safety is the most important thing for my project when the drone is in use, as I do not want to risk injuring members of the public. Level 3 is the “legal” implications. I should not break any laws in the use of the drone; this is why my project is focused on an indoor environment to avoid staying into protected airspace. The last level relevant to me is the 4th level “Privacy”, it is important to take people privacy into account when using the drone.

When in use the drone will not store images of other people without the express permission of the person in question.

The biggest concern of news agencies and members of the public over drone use is privacy. Although a drone's main task could be totally unrelated to capturing data, it could still be outfitted with sensors and cameras to gather photos and data as it flies. Amazon Drones that are currently in development for fast deliveries in the USA have raised concerns over whether they would also be used to capture data. An article on truth-out raises speculation that commercial drones could be used to gather data [2]. This is very unethical, but as regulations are being introduced on drone use, the public could be easily safeguarded from this to prevent corporate entities from saving or using data gathered by drones other than that which is used for the drone to operate safely.

1.2.2 Ethics and legalities of my project

There are legal implications of flying drones in the United Kingdom. These are set out in law [3]. The relevant article for small aerial aircraft is set out in part 22, article 166. You can see in Figure 1 the laws that are laid out in regards to small unmanned aircraft. Also article 167 applies to the drone I am using, as it has a camera built in to the body, see Figure 2.

Small unmanned aircraft

166. (1) A person must not cause or permit any article or animal (whether or not attached to a parachute) to be dropped from a small unmanned aircraft so as to endanger persons or property.

(2) The person in charge of a small unmanned aircraft may only fly the aircraft if reasonably satisfied that the flight can safely be made.

(3) The person in charge of a small unmanned aircraft must maintain direct, unaided visual contact with the aircraft sufficient to monitor its flight path in relation to other aircraft, persons, vehicles, vessels and structures for the purpose of avoiding collisions.

(4) The person in charge of a small unmanned aircraft which has a mass of more than 7kg excluding its fuel but including any articles or equipment installed in or attached to the aircraft at the commencement of its flight, must not fly the aircraft—

(a) in Class A, C, D or E airspace unless the permission of the appropriate air traffic control unit has been obtained;

(b) within an aerodrome traffic zone during the notified hours of watch of the air traffic control unit (if any) at that aerodrome unless the permission of any such air traffic control unit has been obtained; or

(c) at a height of more than 400 feet above the surface unless it is flying in airspace described in sub-paragraph (a) or (b) and in accordance with the requirements for that airspace.

(5) The person in charge of a small unmanned aircraft must not fly the aircraft for the purposes of aerial work except in accordance with a permission granted by the CAA.

Figure 1 - The Air Navigation Order (2009) Article 166

Small unmanned surveillance aircraft

167. (1) The person in charge of a small unmanned surveillance aircraft must not fly the aircraft in any of the circumstances described in paragraph (2) except in accordance with a permission issued by the CAA.

(2) The circumstances referred to in paragraph (1) are—

(a) over or within 150 metres of any congested area;

(b) over or within 150 metres of an organised open-air assembly of more than 1,000 persons;

(c) within 50 metres of any vessel, vehicle or structure which is not under the control of the person in charge of the aircraft; or

(d) subject to paragraphs (3) and (4), within 50 metres of any person.

(3) Subject to paragraph (4), during take-off or landing, a small unmanned surveillance aircraft must not be flown within 30 metres of any person.

(4) Paragraphs (2)(d) and (3) do not apply to the person in charge of the small unmanned surveillance aircraft or a person under the control of the person in charge of the aircraft.

(5) In this article 'a small unmanned surveillance aircraft' means a small unmanned aircraft which is equipped to undertake any form of surveillance or data acquisition.

Figure 2 - The Air Navigation Order (2009) Article 167

For the purpose of my project I will be able to comply with these laws as I am only going to be investigating autonomous navigation in an indoor environment. I will do so in a large open room, with minimal spectators present. For privacy implications I will ensure the drone does not record in normal operations and if set to record for evidential/promotional purposes that anyone included in the filming has given express written permission.

In future developments of the project beyond the scope of this dissertation, outdoor flight may be possible within these regulations but further investigation will be needed in regards to fully autonomous flight outdoors and the regulations surrounding autonomous outdoor flight and any permission that may need to be obtained from a military prospective. Also outdoor flight would provide issues with ensuring that the aircraft is not within 50m of any person.

1.3 Motivations

My motivations on this project stem from an interest into artificial intelligence and the use of autonomous robotics. Futuristic science fiction films constantly use depictions of intelligent robots to perform autonomous tasks without input from the user, and while many of these films depict robots becoming self-aware and turning on their creators, I have an interest in what practical applications the technology could have on actual everyday life.

The principles behind the project could also have many positive impacts on current society. From autonomous search and rescue robots, to autonomous robots working in situations that could otherwise be extremely dangerous to human beings. While the applications

could also be turned to warfare, which is a highly debated ethical issue, there are many positive uses for the technology.

1.4 Work Breakdown Structure

A work breakdown structure will allow me to break down my workload into manageable chunks and levels. My work breakdown structure is shown below in Figure 3, also attached in the appendix in higher quality (8.7 Work Breakdown Structure).

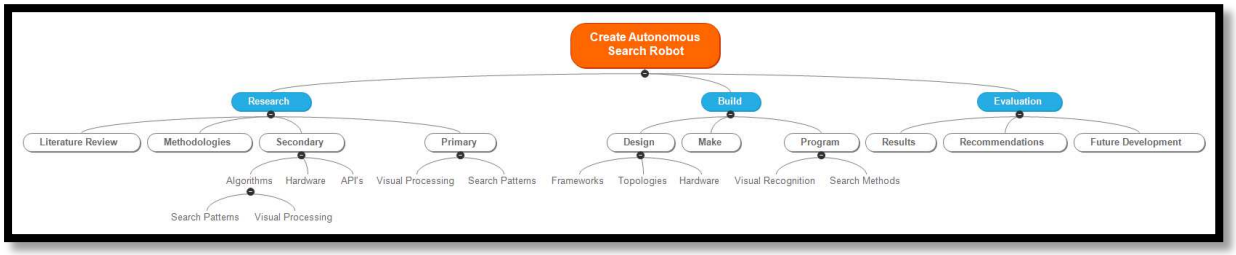


Figure 3 – Work Breakdown Structure

1.5 Risk list

I have created a risk list to allow me to manage and mitigate the risks involved in my project. This risk list covers risks to the project as well as safety implications of my research. Each risk has a magnitude, calculated using the probability of the risk happening and the impact of such a risk. For each risk I have outlined a way of mitigating the risk presented to reduce the risk of it happening.

For my risk list please see the appendix (8.8 Risk List)

1.6 Scheduling

Scheduling is important to my project to outline what tasks I should be doing and when they should be conducted. Scheduling helps keep a project on schedule and prevent it from overrunning project deadlines and budgets.

For my project schedule please see the appendix (8.6 Project Schedule)

2.0 Literature Review

2.1 Introduction

For the purpose of my project I should review as much current work into the field that my project is in. This will help to not only give me ideas for my own project, but to also see how other people have conducted similar research so that I do not need to duplicate research unnecessarily or waste time pursuing an area of research which has already been found to be impractical. The review will give me a greater understanding of the subject area and allow me to identify gaps in current research.

My review will look at papers published in the field but also look at some practical implementations that are currently being worked on that may not have a paper associated with them.

2.2 Reviews

2.2.1 ArduCopter

ArduCopter is a software system designed to work with a variety of Ariel vehicles, including drones [4]. ArduCopter is a UAV control software that allows helicopters and multi-rotor helicopters to be flown automatically. It does this by using GPS and waypoints that the user sets on a map. The software also boasts other automation tasks such as helping the copter hover or “loiter” in an area.

Some of these automated tasks may be of interest to me, although the software is automated, it is not autonomous. The user still sets the locations for the drone and commands such that you would give an autopilot system. For the purpose of my project I am not telling the drone its destination, but letting it decide itself.

2.2.2 Amazon Prime Air

Online shopping company Amazon have started to develop some technology in their research and development department that involves the use of multi-rotor copters, called Amazon Prime Air [5]. The idea is that the copter will lift a small package and deliver it to its destination within 30 minutes. This is technology in its very early stages, and the company has not yet released much information on how the drones will work.

The drone could work in one of two ways, either being piloted by a human who is located back at the drones headquarters, or by use of autonomous navigation and having the drone fly itself to a given destination, land and then return to base.

Either way presents difficulties in both range and also dealing with the elements in the environment around it. For the purpose of my project I am most interested in the autonomous method.

2.2.3 Autonomous Navigation, object detection, and retrieval in an indoor environment

This paper presented in 2012 [6] is the closest piece of literature I have found to what I am attempting to accomplish. The paper was written as part of mission 6 of presumably a robotics competition. The team used an AR drone to autonomously fly into a room through a 1mx1m window, search for a flash drive, retrieve it and then exit through the same opening.

The team use 4 SONAR sensors to allow the drone to map its position in a room, using the telemetry from the sensors the drone creates a 3 dimensional cube and then constantly updates this map when the drone moves as well as storing its traversed path. The AR drone built in camera is used for image processing to act as the drones vision. In total the AR drone as outfitted with "Xbee, Microcontroller, 2 servos, Magnetic pickup mechanism and 4 SONARS" [6]. Some of these additions will be of interest to me in designing my project test prototype.

They use a driver called "The AR Drone Brown Package" which as part of the "Robot Operating System" Library [7]. This allows them to give commands to the drone to allow it to move as well as get back telemetry and camera images.

The system is split into two components, the first is the drone and the attached sensors all connected to and Arduino. The second component of the system is a cluster of computers that does all the computational processing of the data and issues commands to the drone over the Xbee wireless connection. Figure 4 shows the papers overall architecture for doing this, the left side shows the Custer of computers and how they are connected and the right side shows the drone and the connected sensors.

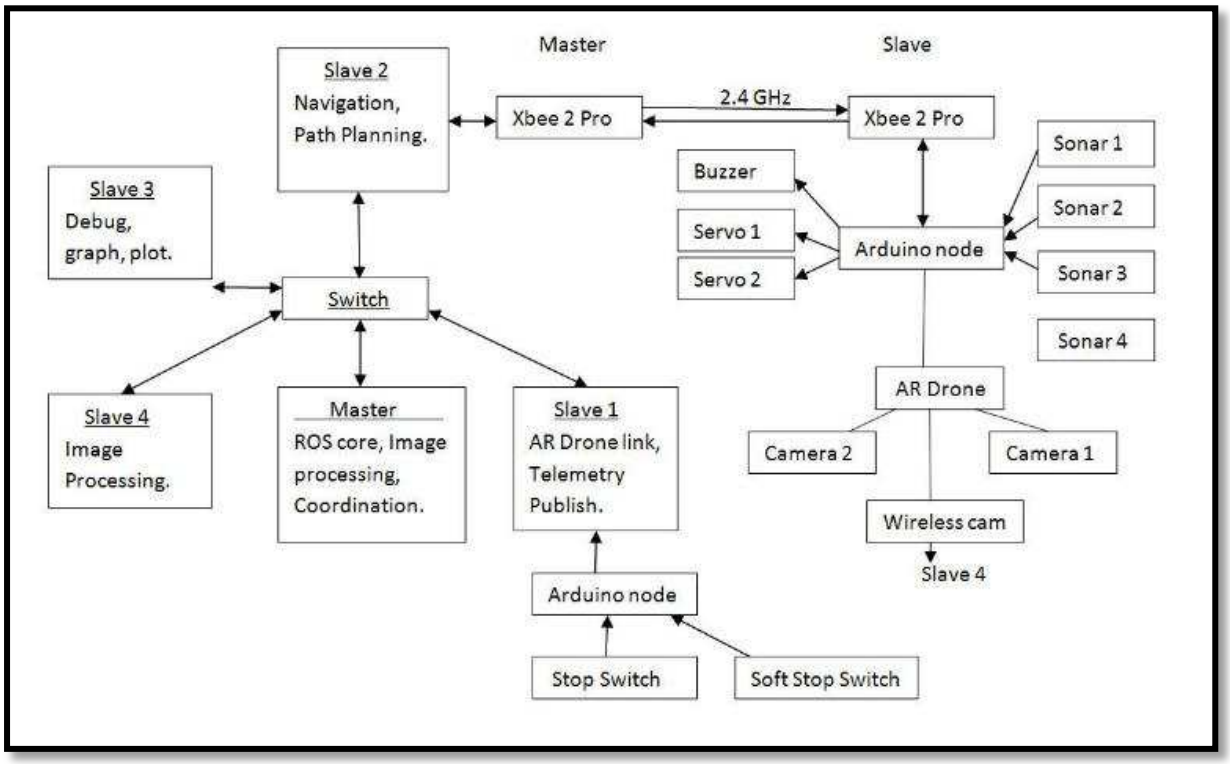


Figure 4 - Overall System Architecture [6]

The paper also references OpenCV [8] which is an open source library for computer vision and machine learning. This library may be of much interest to my project as it contains over 2500 algorithms for vision and machine learning.

2.2.4 Framework for autonomous on-board navigation with the AR.Drone

This paper presents "a framework for autonomous flying" [9]. The framework uses an AR drone as its flying platform and all the navigation is done on board the attacked microcontroller compared to the previous paper which did its calculation on a cluster of computers [6].

Figure 5 shows the system architecture used by this paper for controlling the drone. The drone is interfaced with a microcontroller that if firmly attached to the drone, the microcontroller processes the data from the drone's sensors and plans a path before sending commands back to the drone for flight. It also shows the option for a base station to allow the user to select a path and to log data.

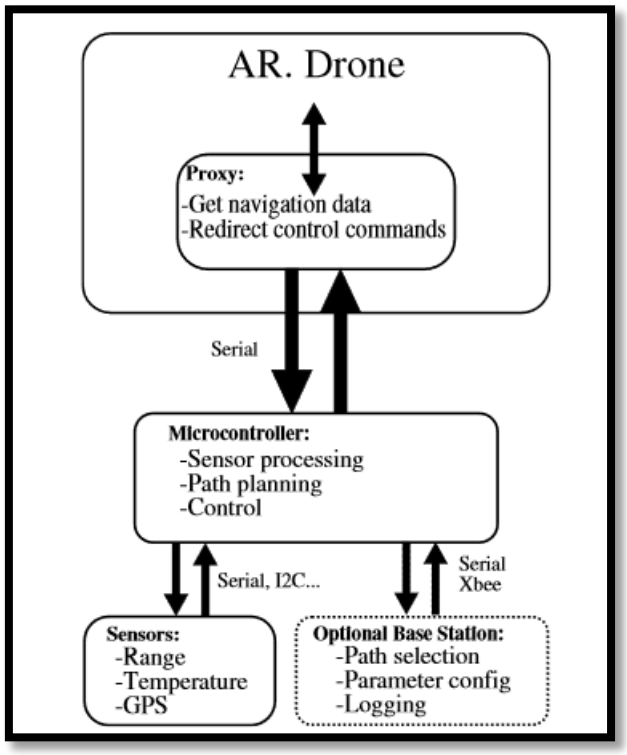


Figure 5 - System Architecture [9]

They attempted to use as little hardware as possible to be cost effective and keep weight down. The on board sensors where used to allow the drone to use estimation algorithms to work out where it had travelled and to allow it to adjust its course to continue to follow the path set.

Experiments from the paper show the path that they intended to follow, the path the drone took, and the path that the drone estimated it was taking. The estimations where surprisingly accurate, of about 50cm which is very good for an aerial platform.

The method of doing all processing on an on board processor is very interesting to me for my own project as it would cut down on cost and the need for extra wireless communications between a base station and the drone.

2.2.5 Low cost MAV platform AR-drone in experimental verifications of methods for vision based autonomous navigation

This paper presents a different method of navigating using the AR drone. The methods presented “rely on visual navigation and localization using on-board cameras” [10], which means all the navigation is done using computer vision with a camera rather than any SONAR sensors.

The experiment conducted works in two parts, to start off with the drone is remotely operated though an outdoor environment to a desired location, the environment is mapped

during this flight using the on-board camera and the in the second part of the experiment the drone is flown fully autonomously using the data it collected in the first flight.

Other experiments conducted also use visual navigation to visit “areas of interest” in an area inaccessible to humans or ground robots. Then drones were also used in an experiment where multiple drones where flown at once in a search and rescue scenario to find “victims” while also avoiding obstacles in the environment.

The visual navigation used on this paper is intriguing though probably beyond the scope of my project for the use in both navigation and object identification. Although if time allowed then using the camera to detect possible obstacles would be an advantage.

2.2.6 Visual tracking and following of a quadrocopter by another quadrocopter

This paper presents a “follow-the-leader scenario” [11], in which a quad copter follows another quad copter (AR drone). The paper proposes using Infrared beacons on the drone to be followed and using an Infrared camera on the following drone.

The following drone uses these infrared beacons to symmetrically follow the other drone and to also estimate its distance from the drone it is following. The infrared camera is mounted to some servos that allow it to pan and tilt to see the drone it is following even when it is not directly ahead.

This method of following the leader could be a possible research path for me; I could adapt the methods to use an infrared heat sensor to pick out human beings to either avoid contact but to allow the drone to enter a follow mode to follow the user/researcher around the indoor space.

2.2.7 Autonomous indoor helicopter flight using a single on board camera

Like a previous paper this particular paper uses “an on-board light-weight camera as the only sensor” [12]. Unlike many of the other papers I have read this one does not use an AR drone as its test platform, but instead uses a normal toy helicopter adapted to carry a camera.

In the experiments conducted a ground robot is first used to navigate the environment and build up a database of images for the helicopter to use when it is navigating. Learning algorithms are implemented alongside this database to check images from the database with images the copter is receiving through its attached camera.

A computer that is wirelessly connected to the helicopter is used for the image processing and calculations to free up weight on the helicopter itself. The database that is checked

against received images uses “cover trees”, which efficiently allows the algorithm to search for a similar image by branching down the “tree branches” that is most similar.

The paper picks out that using a camera on a flying platform is particularly difficult due to the wobble created from vibrations and air turbulence. This should be minimal on an indoor environment in my own project but I will still need to worry about wobble from the vibrations of the motors during flight.

2.3 Conclusions and Findings

I have made several finding and conclusions from reading these papers that will influence some of my research topic and how I will build my own prototype rig. I like the Idea of using 4/5 SONAR sensors attached to the drone to create a 3 dimensional cube for the drone to navigate within, this would allow me to use some very basic navigation and search patterns with the drones flight while concentrating on image processing to look the block of colour or other object that the drone has been told to search for.

I would like to have all processing done on the drone with the Raspberry Pi that I am attaching; this prevents the need for a base station to control the drone, minimalizing issues with transmitting instructions over Wi-Fi. I would still like to keep a base station that would remotely connect to the Raspberry Pi for remote monitoring of the algorithmic processes and for any emergency shutdowns that need to be implemented.

Some possibilities I have taken from the papers for my own project that could be implemented with time permitting would be using image processing to assist the drone with object avoidance to compliment the SONAR sensors that would be attached to the drone. I would also like to explore the further possibility of using a small heat sensor with the Raspberry Pi as a second camera to detect people for both avoidance of crashing into human beings for health and safety reasons, but to also allow the drone to attempt to follow a selected user.

Through some of these papers I have also learnt about some APIs that I could work with to help with my interaction with the AR drone [7] and also about some databases that are available that provide a vast variety of image processing methods that I could test with the drone [8].

3.0 Research Methodologies

During the process of conducting my research and experiments I will need to devise some methodologies for how I will go about planning the software that I am planning to write, how I will test the algorithms that I am working with, and also how I will go about evaluating the success of my project.

3.1 Software Planning

In order to test the algorithms that I will be using I will need to create a program to run the algorithms with the drone. There is a couple of well-known software methodologies that I could use with my software, as I am testing several methods each test will only be a short life cycle. When I implement the final iteration of the prototype then a longer life cycle will be needed than that of individual algorithms.

The three main project lifecycles that I will look at using are the Waterfall Model, the agile methodology, and the iterative methodology. Each has its advantages and disadvantages in its use for my project. And as my project is unusual in that I am programming multiple small programs a hybrid of these methodologies may be acceptable.

From an overall perspective I will use an agile methodology for the testing of the different algorithms. As such each algorithm to be tested will be a different "sprint" of the project. As each of these sprints is an individual mini-project, then I will use an iterative-waterfall hybrid model. This will allow me to quickly flow through the development of the test algorithm, and iterate over the development section.

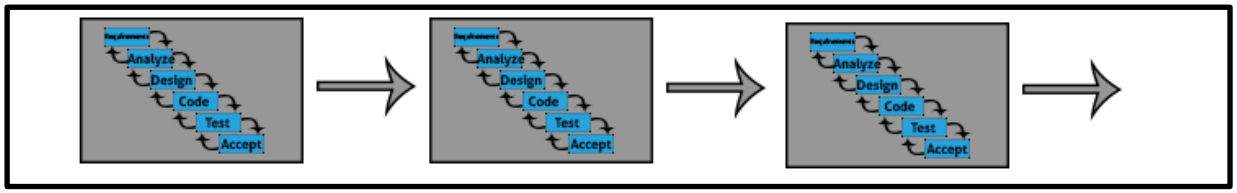


Figure 6 – Hybrid methodology for my project

Above (Figure 6 – Hybrid methodology for my project) is a graphic representation of the hybrid methodology that I have decided to use for my project. Each square represents a sprint of the project, inside each an iterative waterfall for each sprint.

3.2 Algorithm Selection

I will need to come up with a method of testing the algorithms that I am going to be using so that I can make a decision on which algorithm is the best fit for my scenario. I have laid out

some test plans below to help me initially filter out algorithms in my secondary research, and another test plan to then test these final algorithms in practice before deciding on the final algorithm.

3.2.1 Initial Selection Methods

The following guidelines will lay out how I will choose which algorithms from my secondary research I will choose to test practically with the drone.

3.2.1.1 Image Processing

Image processing is a complex task, and there are several ways to identify an object in an image. There are also multiple pre-processing methods that I will need to look at, that deal with making the image more computers friendly.

I will need to consider both in my initial secondary research. And when testing the actual algorithms I will need to make sure the same pre-processing is applied to each one.

I will limit the amount of algorithms to 3, as there are many different variables to be tested with each method. I will need to select these 3 algorithms out of all the ones I find in my secondary research. These should be theoretically bust suited to flight, as images may be choppy, and will also need to be quick. The table below outlines a comparison table to help me decide between the methods.

Algorithm	Advantages	Disadvantages

Figure 7 – Image algorithm comparison table

3.2.1.2 Navigation

There are many different algorithms for navigating a 2 or 3 dimensional space. Some algorithms are dedicated to just navigating a space from A to B, while avoiding obstacles. Other algorithms will move around this space without a final destination, effectively searching the test area.

The algorithms that I am interested in are the search algorithms. As the drone will not know where the object it is trying to find is located, it cannot have a final destination.

For an algorithm to be put through to practice testing it must be a search algorithm, and be able to deal with an unknown test space. To narrow the physical testing I will limit the amount of algorithms to be put forwards to 3. To choose these 3 algorithms I will create a list of advantages and disadvantages for each algorithm, and pick the 3 most beneficial. The table below (Figure 8 – Navigation Algorithm Comparison Table) shows how I will do this.

Algorithm	Advantages	Disadvantages

Figure 8 – Navigation Algorithm Comparison Table

3.2.2 Algorithm Testing Methods

The following sections outline the methods for physically testing the algorithms. These tests will be conducted in a uniform environment, as changing the environment between tests could bias the results of my investigation.

3.2.2.1 Image processing

Below is a table that I have created to record results from testing with the drone. With each algorithm I have initially selected through my secondary research I will test its ability to identify an object. I will do this from different distances and with varying lighting conditions.

Theoretically the best algorithm will be the one that can detect the object from the furthest distance and has the best ability to deal with varying light. The image processing algorithms will initially be tested while the drone is static, and then re-tested during flight.

Algorithm	Test Number	Static?	Distance From Object	Lighting Level (Good/Poor)	Pass / Fail

Figure 9 – Image Test Table

As I will be testing the different algorithms in different lighting conditions I need to define what will be considered a good or bad lighting level. A good lighting level I will consider as the room being bright whit all lights switched on, and a poor lighting level as the room being shaded, with very little lighting turned on, if any at all.

3.2.2.2 Navigation

The below table is similar to the last. I will test several identical scenarios with each algorithm. If it can successfully navigate the room in a searching pattern without hitting obstacles such as walls or posts then it will pass. The best algorithm would theoretically be the one that finds the object in the least time.

Algorithm	Test Number	Obstacles Avoided?	Time Taken to Find Object

Figure 10 – Navigation Test Table

3.3 Evaluation

At the end of the project I will need to evaluate how the final implementation of the project has performed. The project should be able to locate and identify an object, in this case a block of bright pink/purple colour, and it should be able to do this without crashing, and before the drone runs out of battery.

I will also be able to evaluate the individual aspects of the drone's performance in searching and navigating. This will also provide a backup test point for the project, should one aspect of the drones operating needs fail.

4.0 Research

4.1 Secondary Research

4.1.1 Algorithms

4.1.1.1 Image Processing

4.1.1.1.1 A look into image processing

Image processing or image analysis is a wide subject area in the field of computing. There are algorithms which can filter an image, often used for pre-processing to make the image easier to work with. And there are also algorithms that then analyse those images for purposes such as recognition, modelling, motion tracking, shape estimation, segmentation, and the list goes on [13].

Algorithms can also be used for counting a recognised shape or object [14]. Filters are used to smooth images, remove noise, remove colour, morphological filling, erosion, dilation, spatial filters, etc… [15]

4.1.1.1.2 Pre-Processing

The images that I will gather from the drone camera may be of a lower quality to that of a normal every day camera. To improve the chances of one of the algorithms making a match with the object (block of colour) there are several filters I can use to process the image first. A few of these are outlined in this section, Noise reduction, Sharpening, and de-blurring [16].

4.1.1.1.2.1 Noise reduction

When you take a photo you can get what is called noise. Noise can be anything from specs of dust, to a grainy image. Noise can be reduced using a method called averaging [16], this is where the colour for a pixel is altered using an average of the surrounding pixels colours. This helps to blend stray pixels in the image, but over use can cause the image to start to become blurry.

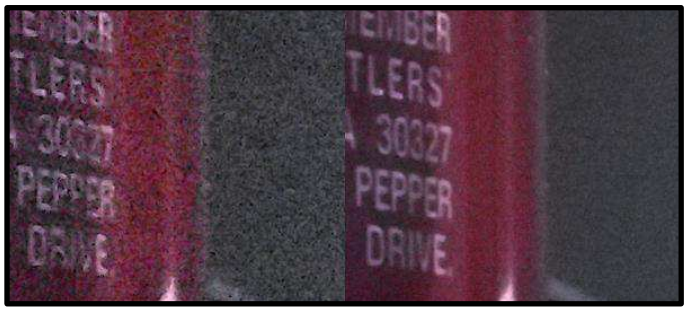


Figure 11 – Noise Reduction [17]

4.1.1.1.2.2 Sharpening

Sharpening is used to enhance details in an image [16]. This is done by identifying the edges in an image and then enhancing them to bring out more detail. The figure below shows how an image can be enhanced with sharpening. Image sharpening can sometimes increase noise in an image.

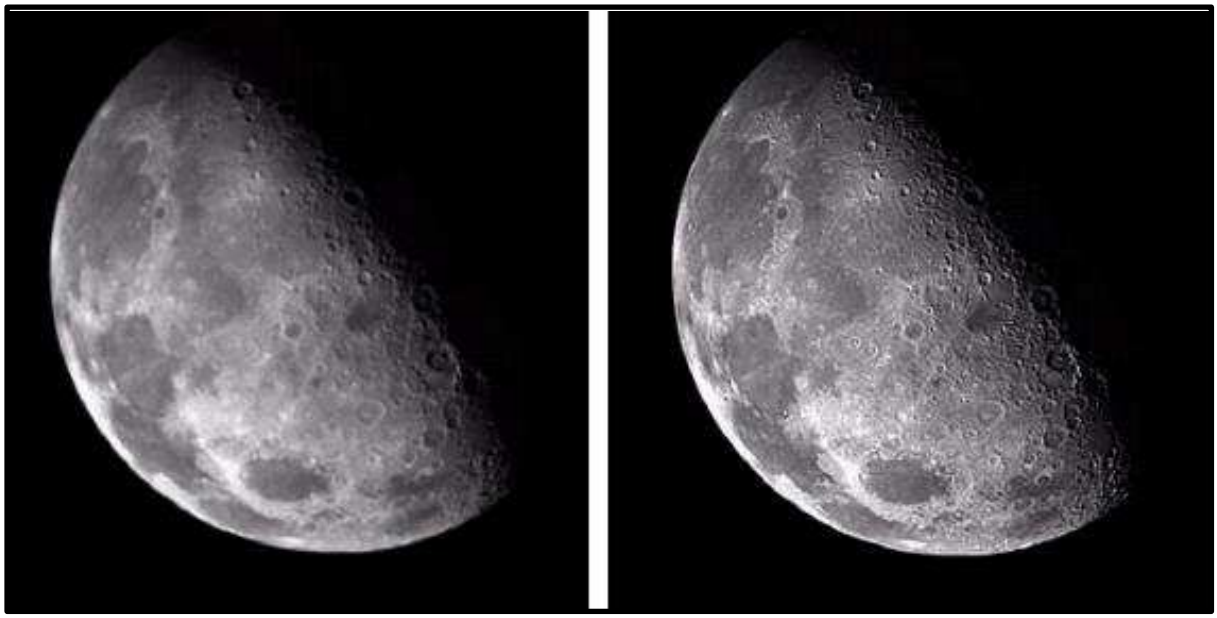


Figure 12 – Sharpening [16]

4.1.1.1.2.3 De-blurring

Blurring of an image is not related to noise [16]. Blurring occurs when the source of the image is moving, causing the image to drag or duplicate the moving object. This will be very relevant to project as the images coming from the drone will be taken during movement; even hovering causes movement due to changes in the air. There are different methods for removing blur; one of them is called constrained least squares filtering, shown below.

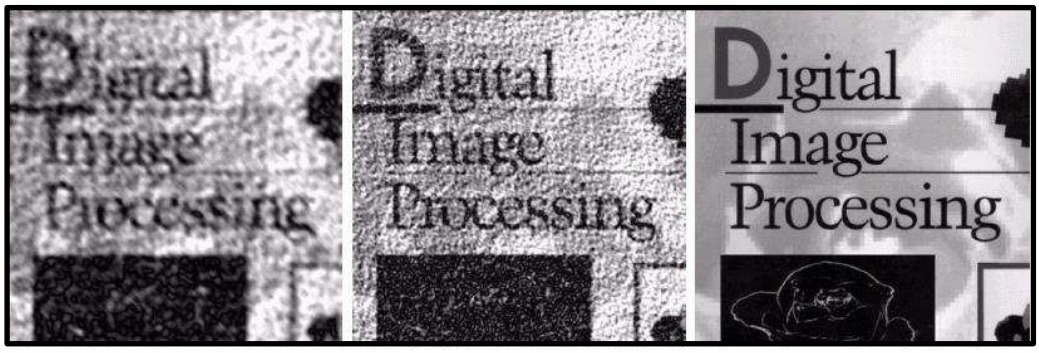


Figure 13 - Constrained least squares filtering [16]

4.1.1.1.3 The Algorithms

These algorithms are the ones which will actually attempt to take the image from the drone camera after it has been processed, and then try and locate if the block of colour that we are searching for is present in the image. There are a limited amount of algorithms for comparing images in the manor required for this project.

4.1.1.1.3.1 Template Matching

Template matching is a promising idea to finding the object in the image. The fundamental idea is finding areas of an image that match a specified template [18] or are similar to the template.

The template is slid around the larger image pixel by pixel and stores a metric based on how similar the images are. At the end of the search the highest matching metric is the area of the image that has matched. If the metric is below a certain level then we can say that the template is not located in the image being compared.

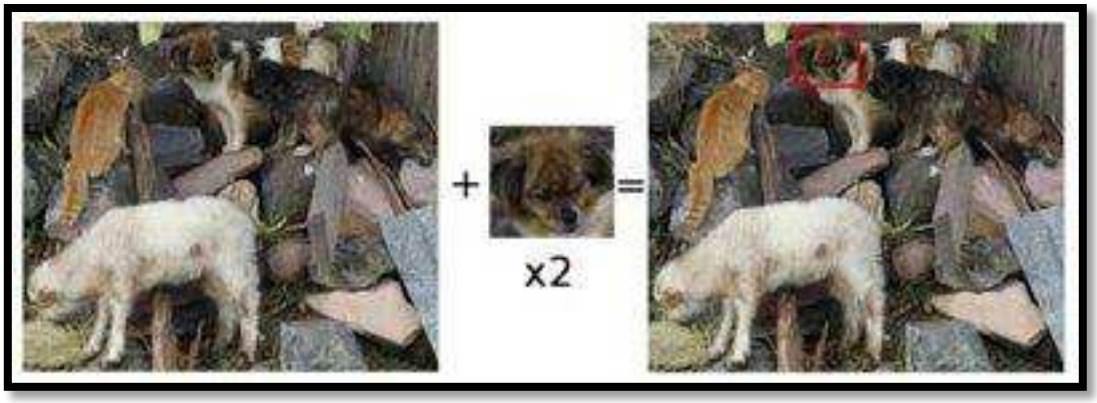


Figure 14 – Template Matching [18]

The above figure shows an example of the two images to be compared. The template is the smaller image in the middle. Multiple templates can be used to help increase the chance of locating the item in the image.

4.1.1.1.3.2 Pixel Classification

Pixel classification works by assigning each pixel in an image to a class [19], splitting up the colours into different classes, such as green for grass, blue for water. You can then find the amount of a type of colour in order to assess how much of that object is in the image.

Colour maximisation can be used to limit the amount of colours in the image. For example RGB maximisation will look at the RGB values of each pixel and change the colour to the highest valued RGB value, (234,256,205) would become (255,0,0).

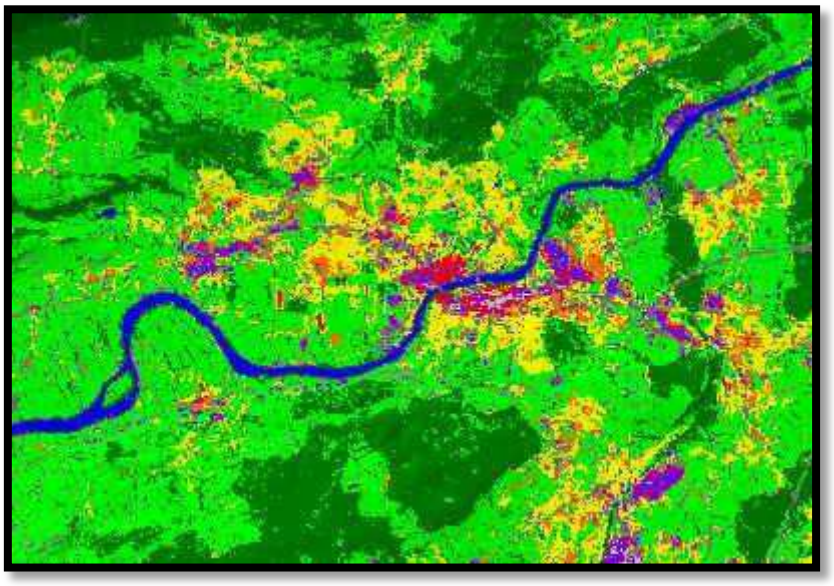


Figure 15 – Pixel classification [19]

In the above image the pixels of similar colour have been changed to an object class. A class represents a range of colour, for example there are many different shades of green, so a greenish pixel will be allocated to the closest colour class.

This method could be very useful for my computer vision project, as it works with colours it would allow me to count up the amount of objects in each colour. If the colour that is on the sheet of paper I am trying to identify (bright pink) is located in a high enough percentage then the object has been found.

4.1.1.1.3.3 Image processing selection

With a limited amount of relevant algorithms that are compatible with the type of computer vision task I am attempting to conduct, I have found that I do not need to narrow down results at this stage, and can test all the algorithms that I have managed to find.

4.1.1.2 Autonomous Navigation

4.1.1.2.1 A look into Autonomous search

Autonomous navigation is used in a wide variety of circumstances, from search patterns for searching an area to path planning algorithms for maps, robots, and games. Path planning algorithms are not useful in the context of this project as they require knowledge of a destination and a map of the room. As we are looking for an object, and don't yet know where it is, we need to use the search patterns. Search patterns are every day methods of searching for something, and can be performed by humans looking for an object.

Most autonomous navigations will have a bias in its movement. This is normally where it has to make a decision on where to go next, for example it could always try to move right first. This is not apparent in search algorithms as the path is pre-defined.

4.1.1.2.2 The Algorithms

The algorithms I will look at may be designed to search a known environment. I.e. the room boundaries are known beforehand. These algorithms can be adapted to deal with an updating world model.

4.1.1.2.2.1 Spiral Search

The first algorithm that I am looking at is the spiral search pattern algorithm. This pattern works by navigating the search space in a spiral pattern until the whole space has been covered. The spiral can work either inwards (from a wall), or outwards (from the centre).

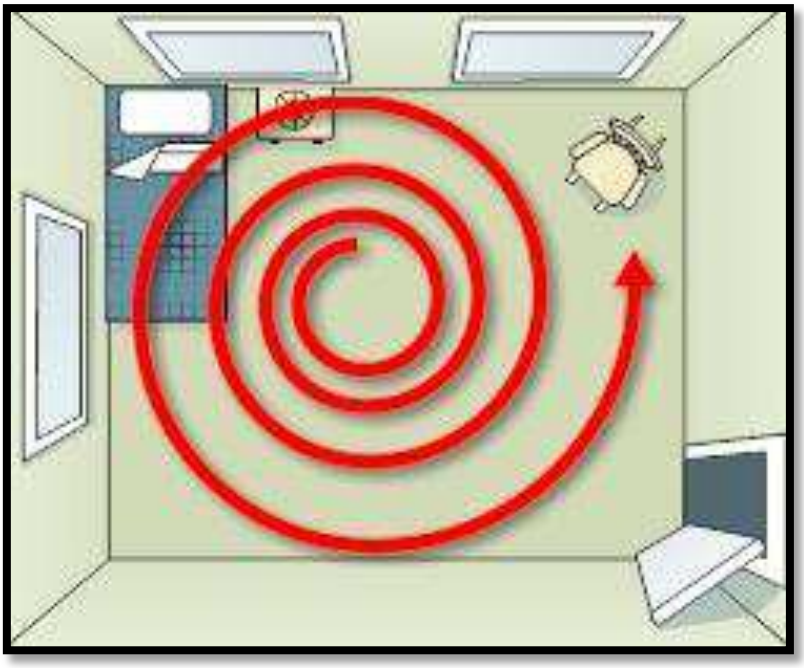


Figure 16 – Spiral Search [20]

The figure above depicts a spiral search in a room starting from the centre and working outwards. A spiral search is a fast and efficient way of covering the entire solution space. Disadvantages can mean that the same area is looked over multiple times when traveling around the spiral.

4.1.1.2.2.2 Random Sampling

The method of randomly sampling has no structure to follow for search. A random direction is chosen by the agent and it then follows that path for a short distance before randomly choosing another direction.

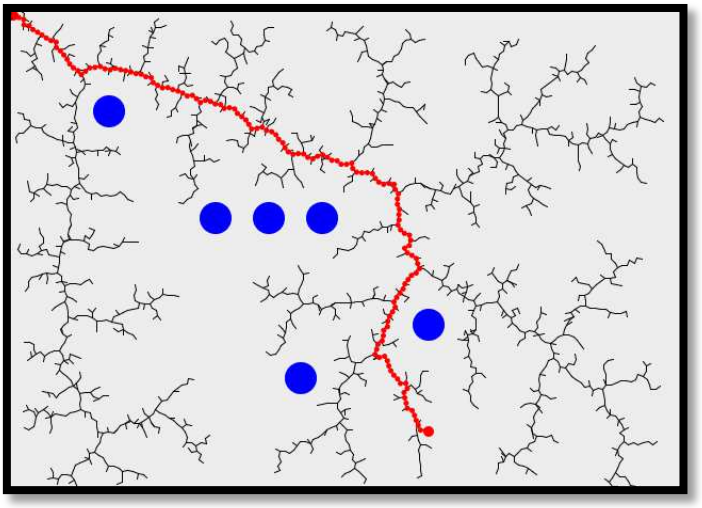


Figure 17 – Random Sampling [21]

The figure above shows one way that a random path finding search may look like. It seems much like the roots of a tree, and then nearing other branches or walls it back tracks to a free spot and branches again. You can also have random paths that just follow around a grid without backtracking.

4.1.1.2.2.3 Grid

Grid search is a simple method involving splitting the search space up into grid squares. The agent can navigate the room by following the lines on the grid, and will eventually cover the entire room. The figure below depicts this search pattern (Figure 18 - Grid Search). This algorithm is potentially very slow as it has much ground to cover, and needs to reset its position on each run of the world. The advantage is that it covers the most space during the search.

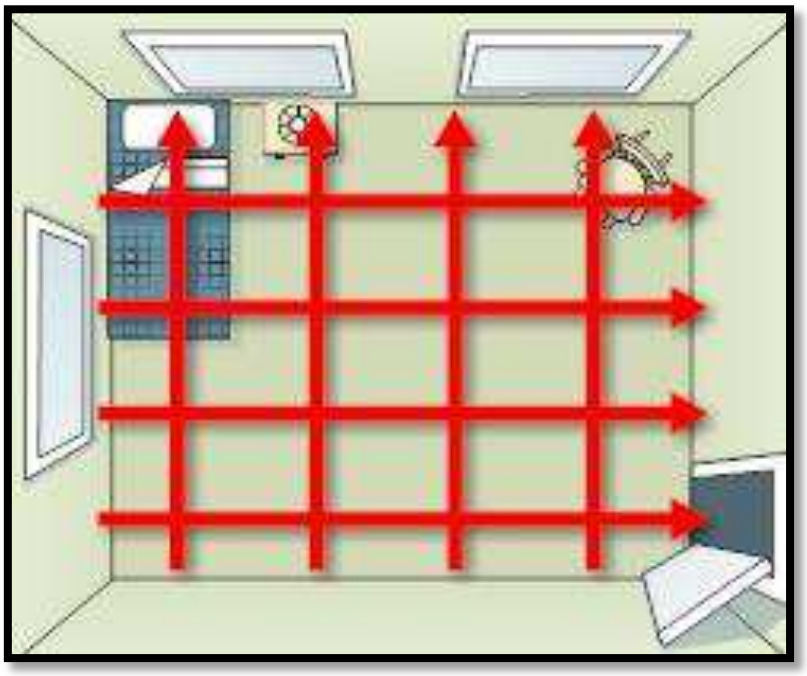


Figure 18 - Grid Search [20]

4.1.1.2.2.4 Zone

The zone pattern is a very simple model. Split into large areas the pattern is extremely quick as it splits down the work space into manageable chunks, without making them too small. A disadvantage is that areas may be missed as the search is not performed in as much detail, however the method may come in use for searching the room with the drone much quicker, as it can see further ahead.

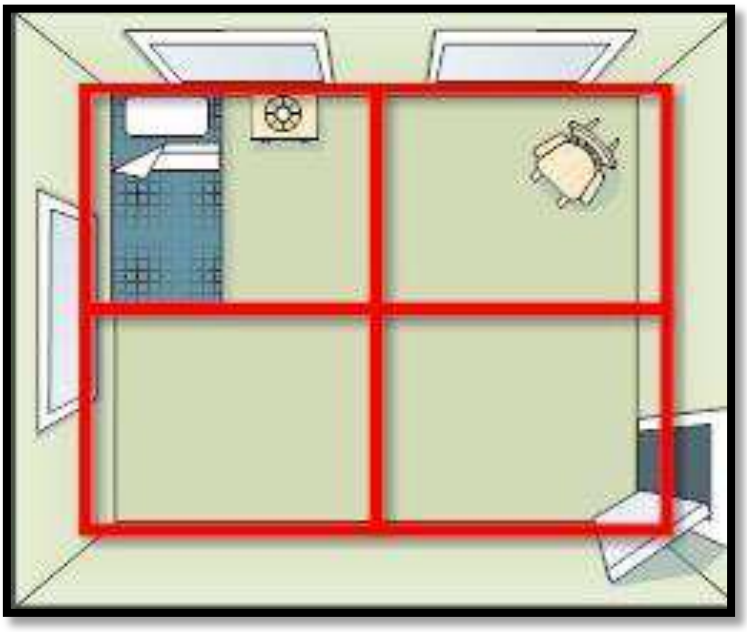


Figure 19 – Zone [20]

The figure above shows the idea of a zone search pattern. The room has been divided into 4 areas to be searched. This will speed up the search time, but may also increase the chance of missing something.

4.1.1.2.2.5 Parallel

A parallel search is very similar to that of a grid search, except the search only heads in one direction. Quicker than a grid search, this method misses out an entire axes from its search area, meaning it probably miss more than the grid, but will be complete in half the time.

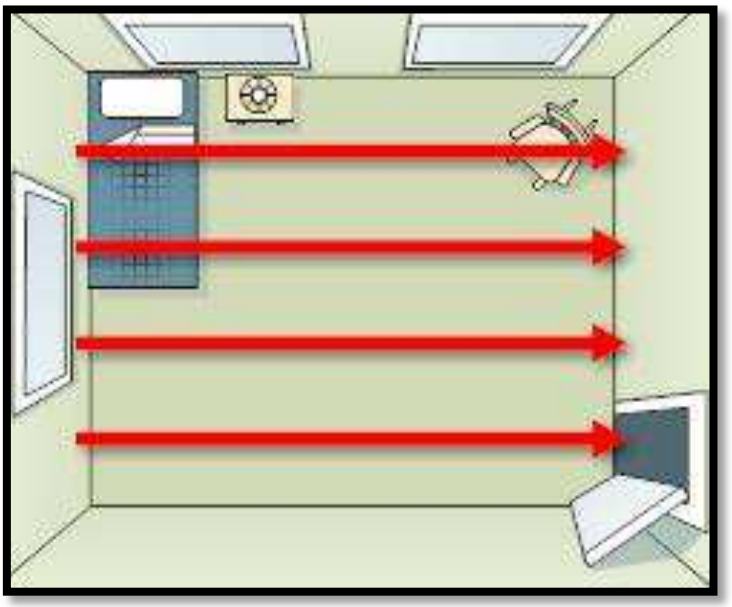


Figure 20 – Parallel [20]

The above figure shows a parallel search for a square room. This may not be a very efficient way for me to search a room for an object with a drone, as the search is only orientated in one direction.

4.1.1.2.2.6 Wall Following

This wall following algorithm will find a wall and then follow it. The advantage of this means it will be able to map out the boundary of the room quicker than the other algorithms. The disadvantage is that following a wall may get you stuck, or may mean walls not attached to the first wall are never explored.

The centre of the search area is untouched by this method, meaning big chunks of search area are being ignored. With the object being searched for in this project likely to be located on a wall, this might not be a bad method to test.

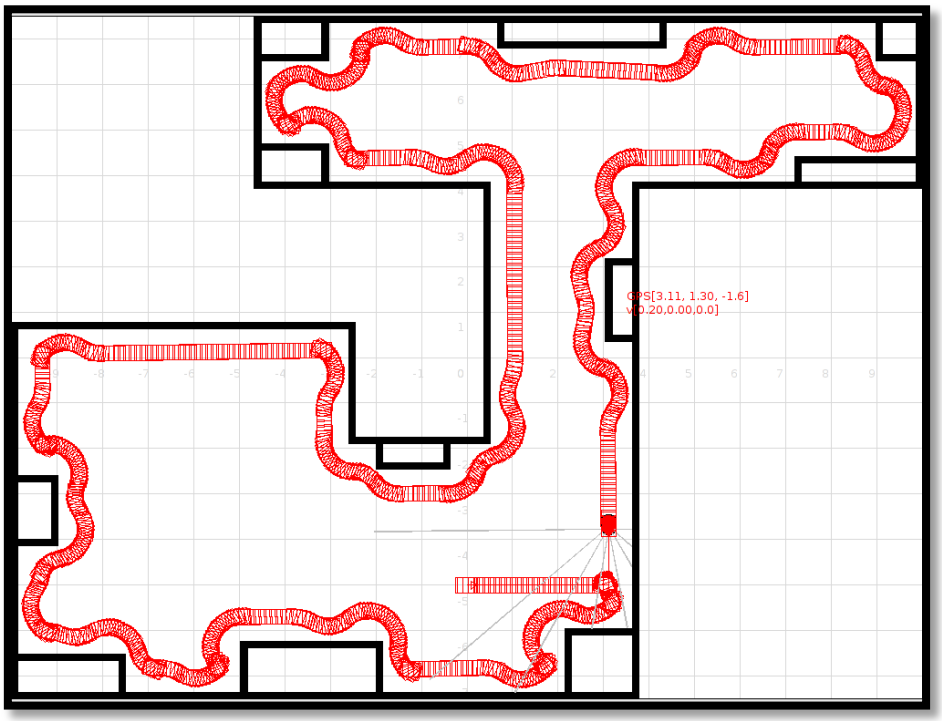


Figure 21 – Wall Following [22]

The figure above shows how a wall following robot made its way to the edge of the room and then proceeded to follow around the edges, avoiding and going around any obstacles.

4.1.1.2.2.7 Search Pattern Selection

As I have 6 different algorithms that can search a room I need to narrow these down to 3 algorithms. The table below shows the comparison of all the search patterns.

Algorithm	Advantages	Disadvantages
Spiral	❖ Large area covered ❖ Quick ❖ Covers all axes	❖ Best if robot is centred ❖ May miss sections of room when encountering walls
Random Sampling	❖ May stumble across object quickly	❖ May take a long time ❖ Covers ground already visited ❖ May never find object
Grid	❖ Large area	❖ Long time to fly entire grid
Zone	❖ Quick –less area to cover ❖ Cuts down search space ❖ Will deal with expanding rooms well	❖ Less accurate as covering larger quadrants
Parallel	❖ Fast, only has a set number of movement to do	❖ Only on one axis, camera may miss object. ❖ Less advanced version of grid
Wall Follower	❖ Visits entire outside of room ❖ Object located on walls	❖ Misses large chunks of the room ❖ May get caught on intermediate walls

Figure 22 – Search Pattern comparison table

Considering the above comparison table I believe the 3 best search patterns to physically test in action are the **Spiral** search, a **Zone** search, and a **Wall Following** search. All the 3 algorithms will cover either a large area, or the main required search area, and will be able to do so in a smaller amount of time in comparison to the other algorithms.

4.1.2 Hardware Capabilities

4.1.2.1 Drone Capabilities

The AR drone has its own capabilities built in that I can use to help control the drone. The main capabilities that I am interested in are listed below

4.1.2.1.1 Camera

This is an important part of the drone for me to have interest in. The drone has two built in cameras on board. I am only interested in the front facing camera at the present time. Using this camera will allow me to implement the visual processing algorithms, without the need to add a camera to the drone and unnecessary weight.

4.1.2.1.2 Sonar

The drone has one built in sonar sensor built into the drone's body. This sensor faces downwards and gives a height measurement for the drone while in flight. This is primarily used by the drones OS to keep it level, but I could use these readouts to change search height.

4.1.2.1.3 Gyroscope

The gyroscope is built into the drone to measure roll, pitch, and yaw. Primarily used by the drone OS for stabilisation, I can use this data to predict if the drone is moving too fast and if the sensor data is being obscured by the drone angle.

4.1.2.1.4 Compass

The compass built into the drone is very important for the navigation of the drone. In creating a virtual 3d environment during run time, the drone needs to know not only where it is in this model, but also which way it is facing.

4.1.2.1.5 Wi-Fi

The drone communicates with its controllers via a Wi-Fi connection. Id does this by booting up its own wireless network and acting as a router, which devices can then connect to, to control the drone.

4.1.2.1.6 Weight Limit

This is another important aspect of the drone. This will greatly affect my project, as to navigate the drone I am intending to add a raspberry pi, with 4 sonar sensors. The drone weighs about 420g with an indoor hull [23]. AR drone enthusiasts [24] suggest the weight with the indoor hull is more like 432g, and suggest an optimal max weight of 480g, although there are no actual listed weight limits for the drone.

4.1.2.2 Extra Hardware Available

4.1.2.2.1 Arduino

Another board that I could use with the drone is called the Arduino. This board is well established and well used with much documentation. It is designed to be easy to prototype products with, without the need to solder together components. My other option for the control board is a Raspberry Pi.

4.1.2.2.2 Raspberry Pi

I have access to a Raspberry Pi for my project. I have chosen this as the board for use with the sensors that I will use for my project over other boards such as the Arduino. This is because I had some Raspberry Pi's easily at my disposal, but also with the launch of the Pi receiving a high level of media coverage; it would be a nice way to test out the new technology.

The following diagram shows the pin schematic of the raspberry pi [25]. There are 8 GPIO pins that are located on the Raspberry Pi model B. These are the ports that I will need to work with for the input and output of the sensors.

Raspberry Pi Model B

3.3V	1	2	5V
I2C1 SDA	3	4	5V
I2C0 SCL	5	6	Ground
GPIO 4	7	8	UART TXD
Ground	9	10	UART RXD
GPIO 17	11	12	GPIO 18
GPIO 27	13	14	Ground
GPIO 22	15	16	GPIO 23
3.3V	17	18	GPIO 24
SP10 MOSI	19	20	Ground
SP10 MPSO	21	22	GPIO 25
SP10 SCLK	23	24	SP10 CEO N
Ground	25	26	SP10 CE1 N

Figure 23 – Raspberry Pi Pin Diagram [25]

The raspberry pi is powered by a micro-USB port. As the drone has a built in USB port for saving images and video to a memory stick, I can use this USB port with the Pi to provide power and save weight.

4.1.2.2.3 Sensors

4.1.2.2.3.1 SONAR Sensor

The HC-SR04 Ultrasonic Module is my chosen sonar sensor for use with the Raspberry Pi [26]. The sonar sensor has 4 pins, these are +5v, ground, trigger, and echo. The trigger pin is activated to send out an ultrasonic pulse and the echo pin is used to listen to the echo that is returned by that pulse.

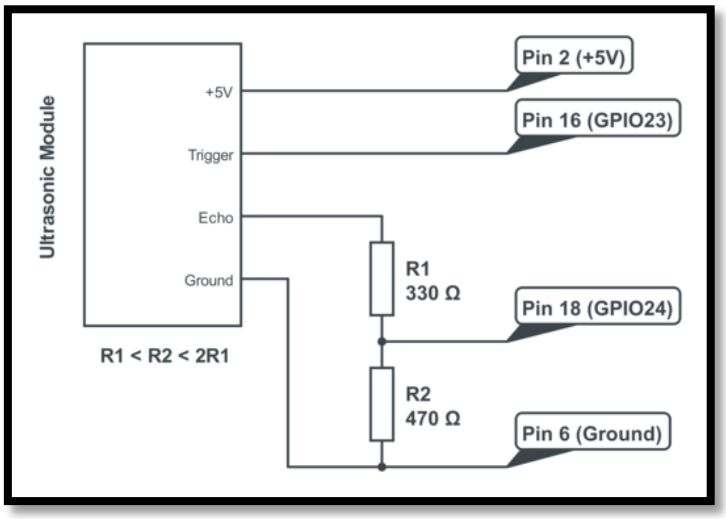


Figure 24 – Sonar Sensor Schematic [27]

The above schematic shows the pin layout for the sonar sensor for use with the Raspberry Pi, along with the required resisters to prevent an overload of power to the sensor [27]. As the Raspberry Pi has 8 GPIO pins, I can use no more than 4 of these sensors.

4.1.2.2.3.2 Heat Camera

Raspberry Pi "NoIR" [28] is an infrared camera for the Raspberry Pi that can be plugged into the camera port on the Pi circuit board. I could use this in conjunction with the sonar sensors on the Pi, as it has no need to use the GPIO pins that are needed for the sonar sensors. The camera would allow me to see the IR spectrum of light, and combined with the image processing on the drones camera, it could be used by the navigation algorithm to avoid human obstacles.

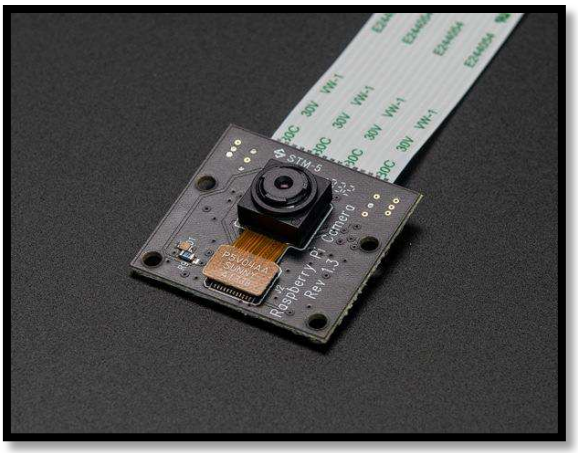


Figure 25 – NoIR camera [28]

For the purpose of this project I will not use an IR camera. While the camera will provide a great addition to the project, it is not absolutely necessary for the operation of the drone and would cause extra weight and power requirements.

4.1.2.2.3.3 IR Distance Sensor

An IR (Infra-Red) distance sensor is another method of gathering distance information for use by the drone. It works in a similar way to the sonar sensor by bouncing IR off of objects and returning the distance based on the amount of IR bounced back [29]. IR sensors are smaller and lighter weight than sonar sensors.

The following figure shows what an IR distance sensor looks like. An infra-red sensor if not really suitable for this project as its object detection range is very limited, and can only sense obstacles up to about 1 meter.

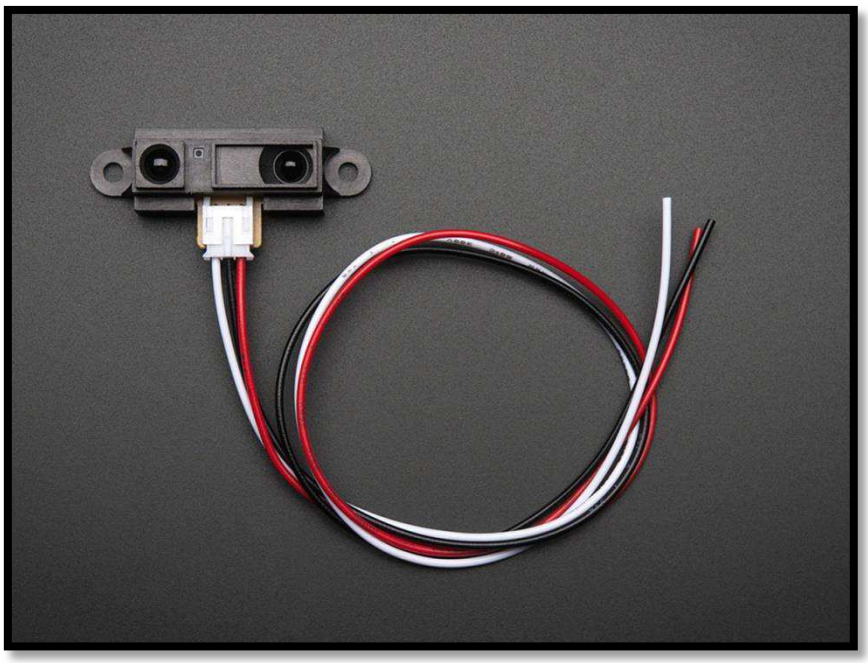


Figure 26 – IR distance Sensor [29]

4.1.2.2.3.4 Distance Sensor Comparison

For sensing distance I can either use a sonar sensor or a IR distance sensor. The sonar sensor is the most accurate and can deal with greater distances, however is heavier and can cost more than an IR sensor. An IR sensor is lightweight and cheap, yet it is not very accurate and cannot deal with great instances. For this reason I have chosen to use the sonar (ultrasonic sensor) sensor, while heavier it will be able to deal with rooms of a greater size.

4.1.2.3 Sensor Configurations

There are multiple ways that I can configure the sensors on the AR drone to provide the best coverage. Also the different search pattern algorithms will require a different amount of

sensors. I have found that I can support up to four sonar sensors on the Raspberry Pi, and with the different search patterns I am testing there are two possibilities of sonar configuration, either one sensor on the front of the drone, or four sensors on each side of the drone.

A one sensor configuration of the drone would save weight and cost. To get a full map of area of a room with one sensor the drone can simply rotate 360 degrees and get the same sensor data as four sonar sensors. This could be a disadvantage as valuable flight time could be sacrificed for constant stopping to rotate and scan the room; it could also mean the drone has the possibility to drift towards an object without realising it. Some of the search patterns such as spiral could manage with one sensor as it does not require a model of the room to move around.

A four sensor configuration will cost more to produce and would be heavier than its one sensor counterpart; however there are advantages in time saving, being able to monitor drift, and the possibility of discovering new areas of a room while performing a search pattern. Search patterns like the wall follower are preferable on this configuration, e.g. the forward sensor keeps the robots distance from the wall, while the other sensors check for junctions and obstacles in its path. For a zone search pattern it would become inaccurate to travel around with one sensor as drift could move it into a different zone on the same axis, a four sensor configuration counters this and it can keep its x and y coordinates at all times.

I will use four sensors for my tests, this is because it will allow be to test all the different search patterns required for my research, however in practice some sensors could be removed. Even with one sensor compatible search methods it is unadvisable to use one sensor as it becomes a near impossible task to deal with drift. The figure below shows the two different configurations of sensors and a visualisation of the drone's field of view.

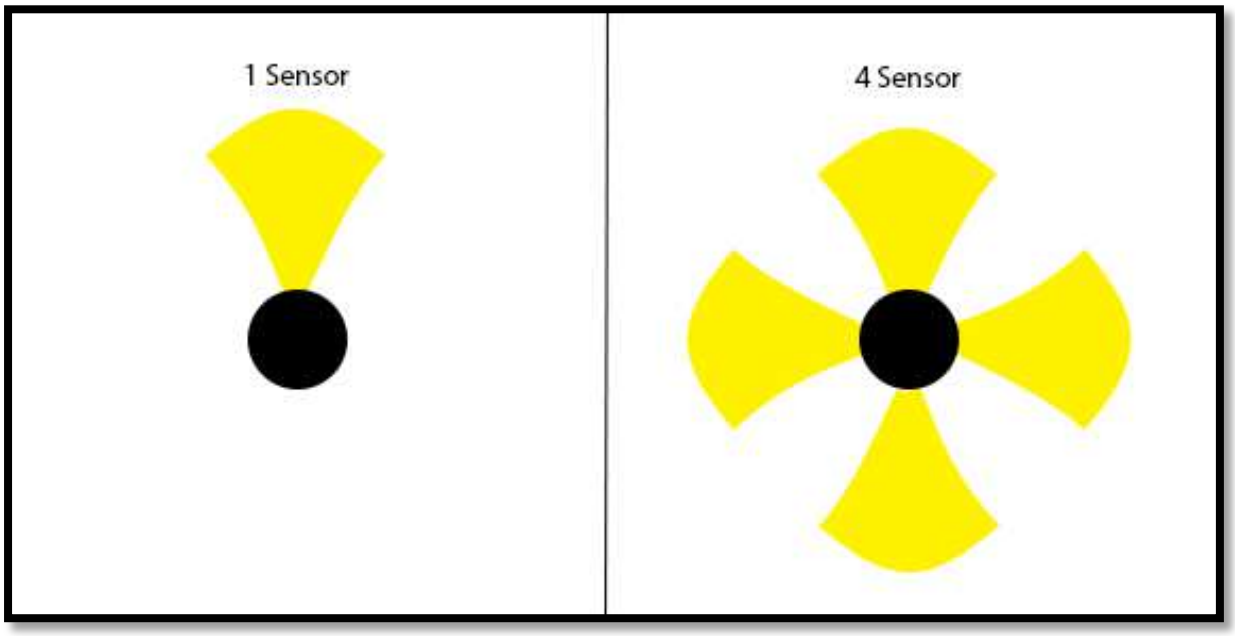


Figure 27 – Sensor Configurations

4.1.3 Software

4.1.3.1 Programming languages used

There are several programming languages that can be used for this project. The different components that I am using each have a preferable language; this is normally the one that the documentation is used with. But items like the sonar sensor being hardware, this code can be ported to different languages.

For the programming I will use both Python and C as they are the two programming languages that the hardware in use operates with in their API's. C will be used with the Drone, and python or C with the sonar sensor.

4.1.3.2 Operating systems

I have a choice of putting several operating systems on the Pi for use with interfacing with the AR Drone. The typical operating system that is used with the Pi is Linux, and has a wide variety of distributions available for it. This would make it a logical choice for OS. The API's that are available for the AR Drone are also primarily available only for the Linux operating system. Because of these two factors it would be unwise to use any other operating system without diverting extra time to porting code between platforms.

4.1.3.3 APIs

There are several API's that are available to me that can make my life easier when programming the project. These relate to the interaction with the drone, and also the different algorithms that I can use.

4.1.3.3.1 AR Drone 2.0 API

The AR drone API will allow me to interface my program with the AR drone and control the flight of the drone, also receiving back video telemetry. The API is well documented online, including a well-documented developer guide [30].

Using this APR prevents the need for me to write my of API interface code, although using the API can be both simple and more difficult at the same time, for example talking off is easy, but moving requires variables for roll, pitch and yaw.

4.1.3.3.2 Node JS Node-Copter

Written in JavaScript using node JS [31], this API could be an invaluable tool for me as it simplifies interfacing and controlling the AR drone greatly. The nodecopter.js also stands for an AR drone programming event.

Commands are simplified to simple function calls, such as connect, take off, or rotate. This removes the hassle of writing my own code to perform these tasks, and allows me to focus on the algorithms, though this package does not provide as much as the API provided directly by Parrot. The API is still heavily under development with little or no documentation or functionality for obtaining video.

4.1.3.3.3 Open CV

OpenCV is an "Open Source Computer Vision Library" [32]. Open CV provides about 2500 different visual processing algorithms. These algorithms do many different things, many of them not relevant to the purpose of this project, but the ones that are would allow me to test the algorithm without the need of programming the visual processing algorithms from scratch.

4.1.3.3.4 Robot operating system

The robot operating system is another API that allows easier access to the AR Drone API [33]. It simplifies commands much like the Node JS Node-Copter API does to single lines of code, and simplified commands like move left, right, etc. Again this would simplify my task as it would remove unnecessary lines of code and allow me to focus on the algorithms that I am testing. The robot operating system is also more difficult to setup as it has a lot more dependencies.

4.1.3.3.5 Drone API Choice

Three of the API's that I have looked at are capable of working with the AR Drone. The original API that is provided by Parrot will be the API that I will use. This is because it has the most functionality for the drone, whereas the others are missing some functionality, also it is easier to set up.

4.2 Primary Research

4.2.1 Testing Platform

The testing platform for the algorithm testing is an AR drone with attached Raspberry Pi and Sonar sensors. The build, and design of this platform is outlined in 5.1 (Prototype Design) and 5.2 (Prototype Building) further on in this report.

For uniform testing I will conduct testing in EC1-29 (The Open Space) of Coventry University. The room is large to allow safe testing and will provide a test space that can be easily configured and changed using display boards.

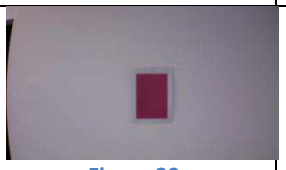
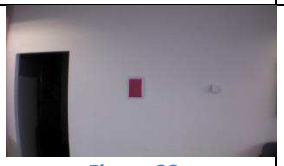
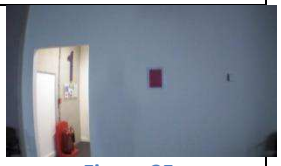
4.2.2 Image Processing

While testing the image processing algorithms I will first test the algorithm in a static position and then test it again while the drone is hovering in flight. The drone will not be navigating during these tests to prevent any bias issues from the navigation, and the drone will be controlled solely by the provided software from parrot [34].

4.2.2.1 Complications

Complications with operating with the AR Drone (5.3.3 Raspberry Pi Issues) have led me to testing the image processing methods with an alternative method. I will fly the AR drone using the provided software and use camera capture to get the images I require for testing. This will continue to be relevant as the images are from the AR drone in the same conditions.

4.2.2.2 Images

Distances	Static (Good Light)	Flying (Good Light)	Static (Poor Light)	Flying (Poor Light)
1m	 Figure 28	 Figure 29	 Figure 30	 Figure 31
3m	 Figure 32	 Figure 33	 Figure 34	 Figure 35
6m	 Figure 36	 Figure 37	 Figure 38	 Figure 39

10m	 Figure 40	 Figure 41	 Figure 42	 Figure 43
15m	 Figure 44	 Figure 45	 Figure 46	 Figure 47

Figure 48 – Table of Images

We can see from the table of images above that the images in flight were not too dissimilar to the images taken while the drone was static. The main differences where with the high and low lighting, although the differences between the lights on, and blinds open to lights off and blinds closed, where not that great. Images will be tested without pre-processing to allow for more accurate results.

4.2.2.3 Template Matching

For template matching I used the OpenCV API. The template matching works by taking a template image and overlaying it over the comparison image. The template image I used was a small block of pink about 8x9 pixels, to allow for recognition where the object seems smaller.

The module code was from the OpenCV tutorials [35], and edited by me to allow me to loop through all the images and test them for matches.

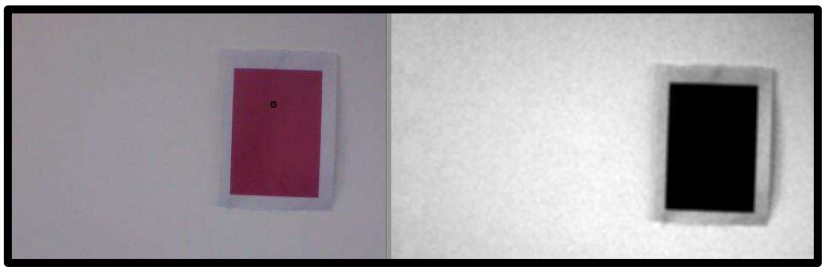


Figure 49 – Template Match Pass

Figure 50 – Template Match Fail

4.2.2.4 Pixel Classification

For pixel classification I was only interested in the pixels that where pink/purple in colour. I looked at the RGB value for every pixel in an image and counted the amount of pixels that matched a RGB range that I had set. If the amount of pixels counted where over the threshold (200 pixels) then I said the object had been found, otherwise the match was not reliable.

To perform this I used OpenCV [32] to load the images into the program and allow me to look at the RGB for each pixel, I then looped through all the pixels and kept a count of any in the pink range. Afterwards I then checked how many there where.

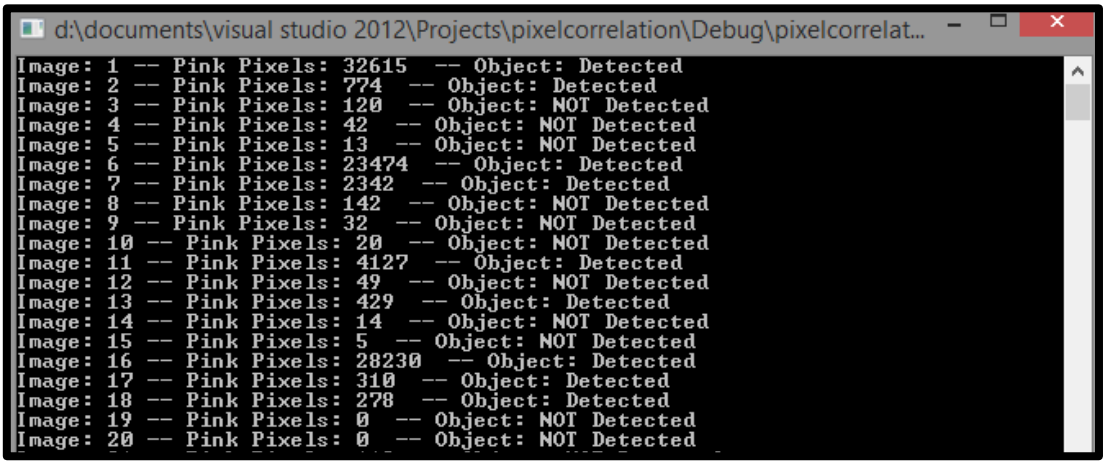


Figure 51 – Pixel Correlation Program

Code for the pixel classification program is attached (8.2.2.2 Pixel Classification). I used the OpenCV documentation to help me create this code [36] and some tutorials from the internet [37].

4.2.2.5 Results

The results table has been altered to integrate with the new testing method. The Static, Distance, and Lighting Level fields have been replaced with an image reference number from the previous section.

Algorithm	Test Number	Image	Pass / Fail
Template	1	Figure 28	PASS
Template	2	Figure 29	PASS
Template	3	Figure 30	PASS
Template	4	Figure 31	PASS
Template	5	Figure 32	PASS
Template	6	Figure 33	PASS
Template	7	Figure 34	FAIL
Template	8	Figure 35	PASS
Template	9	Figure 36	PASS
Template	10	Figure 37	PASS
Template	11	Figure 38	FAIL
Template	12	Figure 39	FAIL
Template	13	Figure 40	FAIL
Template	14	Figure 41	FAIL
Template	15	Figure 42	FAIL
Template	16	Figure 43	FAIL
Template	17	Figure 44	FAIL
Template	18	Figure 45	FAIL
Template	19	Figure 46	FAIL
Template	20	Figure 47	FAIL
Pixel Classification	21	Figure 28	PASS
Pixel Classification	22	Figure 29	PASS
Pixel Classification	23	Figure 30	PASS
Pixel Classification	24	Figure 31	PASS
Pixel Classification	25	Figure 32	PASS
Pixel Classification	26	Figure 33	PASS
Pixel Classification	27	Figure 34	FAIL
Pixel Classification	28	Figure 35	PASS
Pixel Classification	29	Figure 36	FAIL
Pixel Classification	30	Figure 37	FAIL
Pixel Classification	31	Figure 38	PASS

Pixel Classification	32	Figure 39	PASS
Pixel Classification	33	Figure 40	FAIL
Pixel Classification	34	Figure 41	FAIL
Pixel Classification	35	Figure 42	FAIL
Pixel Classification	36	Figure 43	FAIL
Pixel Classification	37	Figure 44	FAIL
Pixel Classification	38	Figure 45	FAIL
Pixel Classification	39	Figure 46	FAIL
Pixel Classification	40	Figure 47	FAIL

Figure 52 – Results Table

4.2.2.6 Conclusion

Both visual classification methods had a similar outcome to each other during testing. The graphical results outlined in the rest of this section show the differences and similarities.

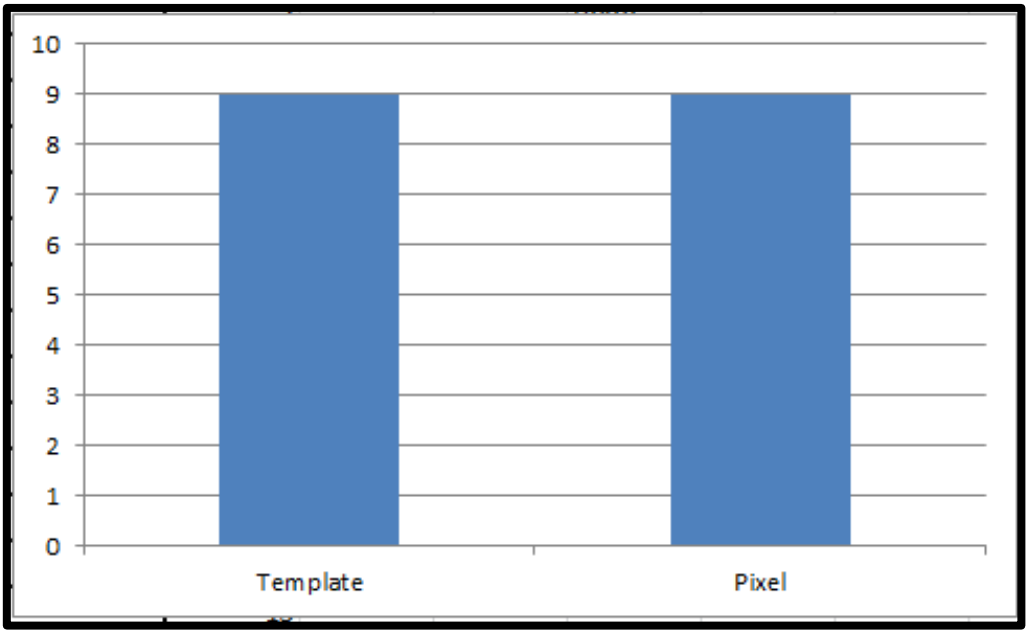


Figure 53 – Overall Pass Rate

The figure above shows the overall results. This is the amount of Passes each algorithm got out of a possible 20. Both algorithms received the same overall mark of 9, suggesting that they are equaly as effective when it comes to finding a block of colour.

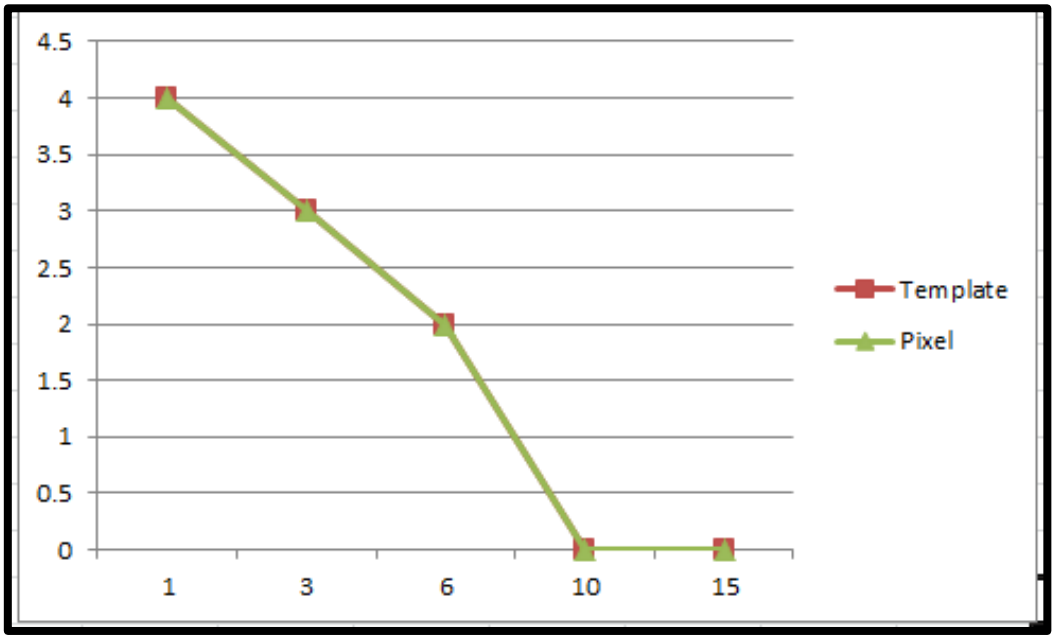


Figure 54 – Distance Pass Rate

In the above figure is a comparison of the amount of passes by distance. Distance is the X axis and amount of passes on the Y axis. We can see that both template matching and pixel classification had the same outcome for each distance. This result also suggests that after 1m the accuracy of identifying the object decreases dramatically. At 6 meters the algorithms are only 50% effective, 75% effective at 3 meters, and 100% effective at 1 meter.

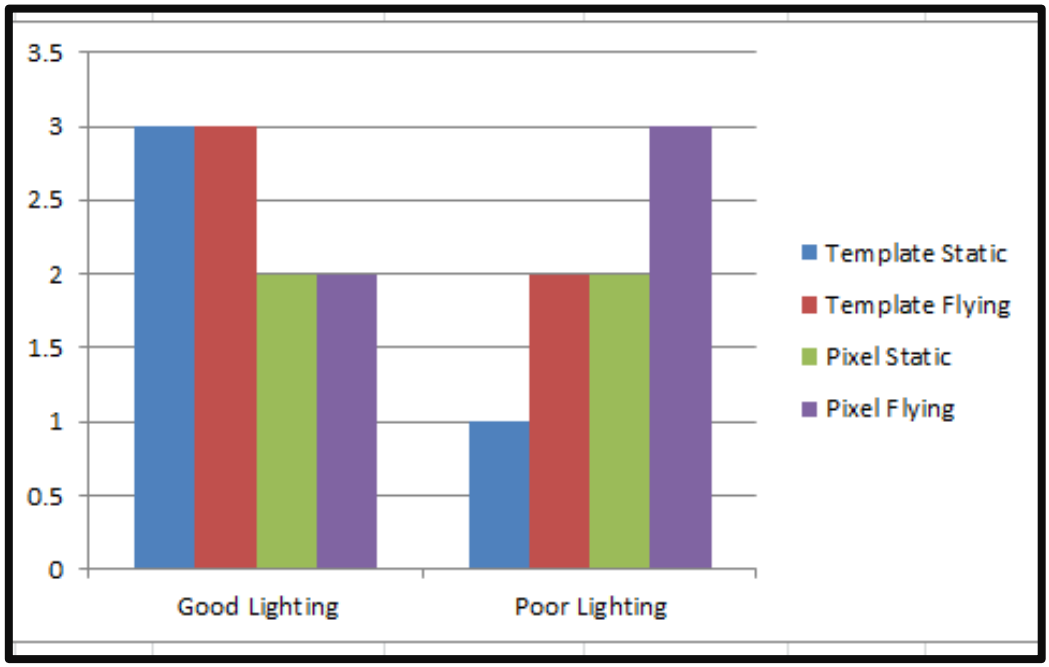


Figure 55 – Flight and Lighting Comparison

The above figure shows the comparison between the lighting conditions and the flight conditions of each algorithm. We can see that in good lighting, template matching was more effective than pixel classification. But in poor lighting pixel classification was better than template matching.

There was little or no difference between a static drone and a flying drone. Pixel classification had a slight increase in Pass rate for a drone that was flying, as well as template matching, both with poor lighting. While during good lighting the pass rate remained the same for flying and static for their respective algorithms.

Looking at the evidence Pixel Classification was better at recognising the object during poor lighting conditions, while Template Matching was better in good lighting conditions. Distance didn't provide much discrimination between the two algorithms with both failing at about the same time, leaving the choice for the better algorithm to the best to deal with the lighting. Overall Template Matching performed better in terms of good and bad lighting, but in overall results the algorithms where equal in their results.

I would consider the Pixel Classification a better choice. It was quicker to generate its results in comparison to Template Classification, and it gave a definitive yes or no answer whereas Template Classification always had an answer, it just either matched the correct part of the image or the wrong part. This lack of a definitive yes or no would be bad for an autonomous system and as such it would deal better with Pixel Classification. Pixel Classification could also be improved by adding more shades of pink to the classifiers and reducing the amount of pixels required to pass the image.

4.2.3 Autonomous Navigation

To limit the range of testing and complexity, I will initially only test the algorithms in a 2 dimensional space by elevating the drone to a constant static height. This will make testing simpler, quicker, and more efficient.

4.2.3.1 Layouts

The following outlines the room layouts I am using for the testing of the search pattern algorithms. The following figure outlines the table of colours.

Colour	Meaning
Black	Wall
Green	AR Drone
Pink	Target Object
Blue	Temporary Wall / Obstacle

Figure 56 – Colour Key

4.2.3.1.1 Layout A

Layout A is a simple open room with no added obstacles. The object is placed in the room at a point that should be easily identifiable. The drone will start in the middle of the room facing away from the object. This layout should be solvable by most algorithms.

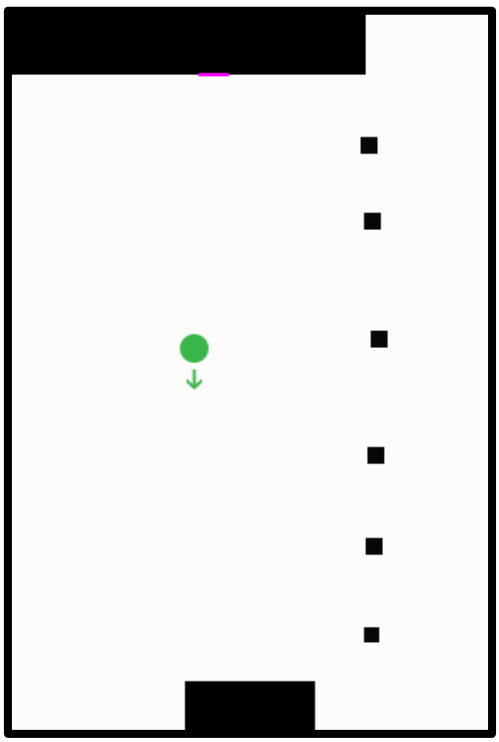


Figure 57 – Layout A

4.2.3.1.2 Layout B

Layout B is more complicated. The drone and object will stay in the same place and an obstacle will be placed between the drone and the object. This will cause one search algorithms to take longer or maybe even not find the object.

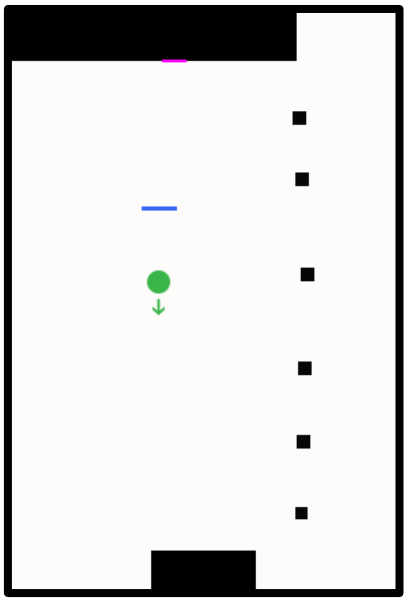


Figure 58 – Layout B

4.2.3.1.3 Layout C

For layout C four obstacles have been placed in the room to "trick" the drone into thinking the room is smaller than it actually is. This will trip up some of the algorithms, but hopefully they should recognise the room opens up as they navigate.

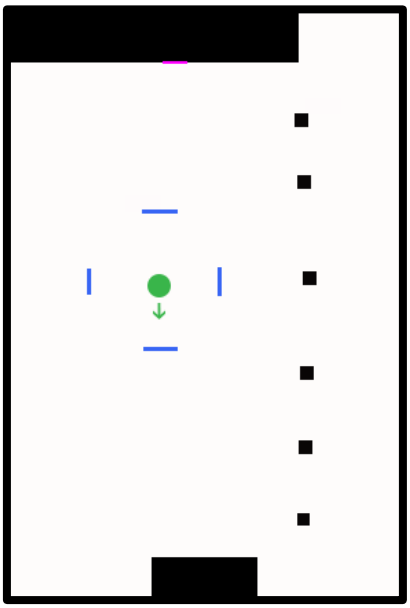


Figure 59 – Layout C

4.2.3.2 Complications

After complications with interfacing the Pi with the AR Drone (5.3.3 Raspberry Pi Issues) I was unable to connect with the drone to get it to fly. To continue researching the effectiveness of these search patterns I will simulate virtually the environment. I will create a 2D grid virtual environment that the virtual drone will have to navigate, each block on the grid will represent 1m and obstacles will be in those blocks. From the visual testing the object will be visible if it is 3m or 3 blocks in front of it, as this was a good distance with 75% accuracy.

To keep the virtual tests simple I will use the same size virtual room and layouts as EC1-29 but remove the columns. Time will be replaced with blocks moved, the more blocks travelled the longer it would take on find the object.

4.2.3.3 Class Diagram

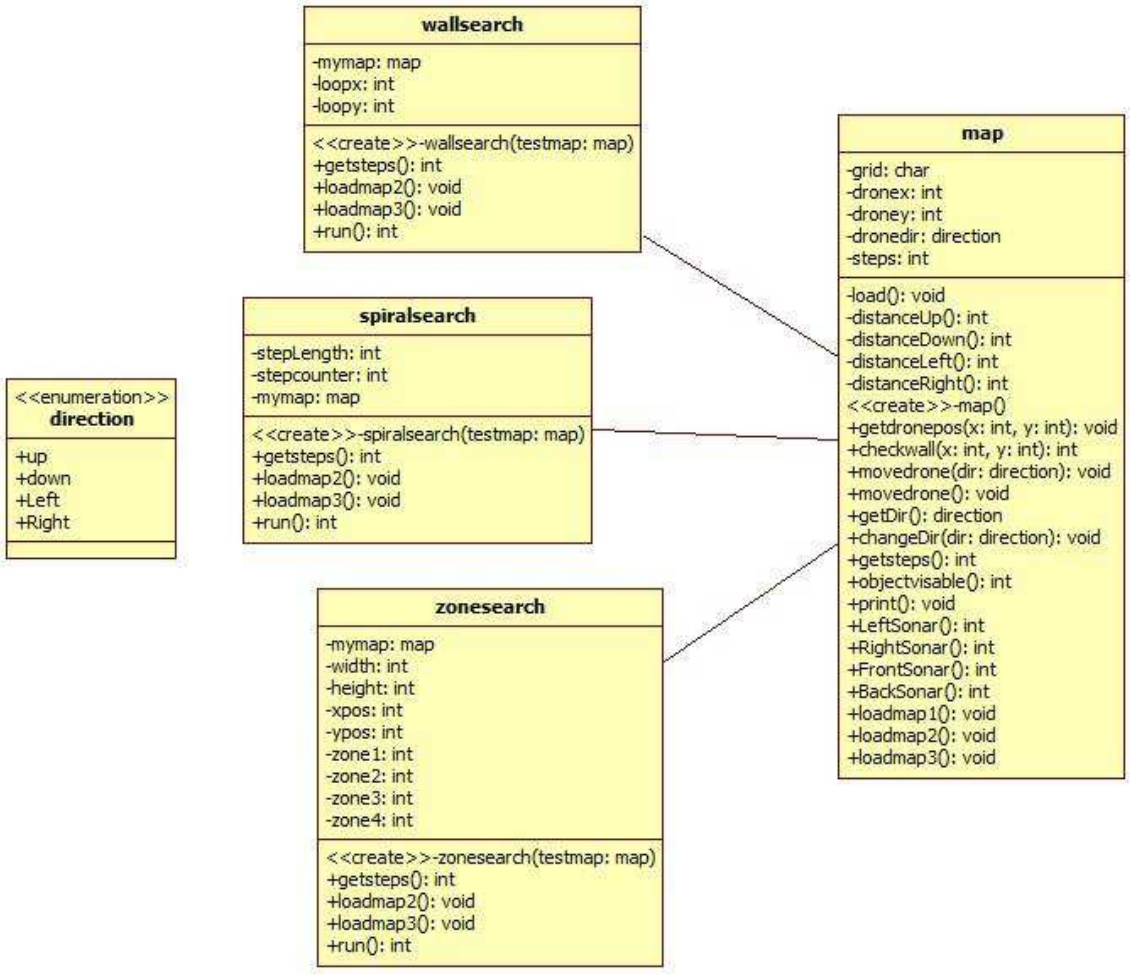


Figure 60 – Class Diagram for search program

The above class diagram outlines 4 main classes for the search pattern testing program. The main class is the map, which is the virtual world that I have created for testing the algorithms, the 3 remaining classes are each their own algorithm.

4.2.3.4 Virtual Layouts

These recreations of the original room layout diagrams show the 2d environment that the virtual drone will be working in. The '#' symbol represents a wall or obstacle that the drone cannot pass through.

23	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
22	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#					#
21	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#					#
20	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#					#
19	#																				#
18	#																				#
17	#																				#
16	#																				#
15	#																				#
14	#																				#
13	#																				#
12	#																				#
11	#																				#
10	#																				#
9	#																				#
8	#																				#
7	#																				#
6	#																				#
5	#																				#
4	#																				#
3	#																				#
2	#										#	#	#	#							#
1	#										#	#	#	#							#
0	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
Y x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Figure 61 – Layout A

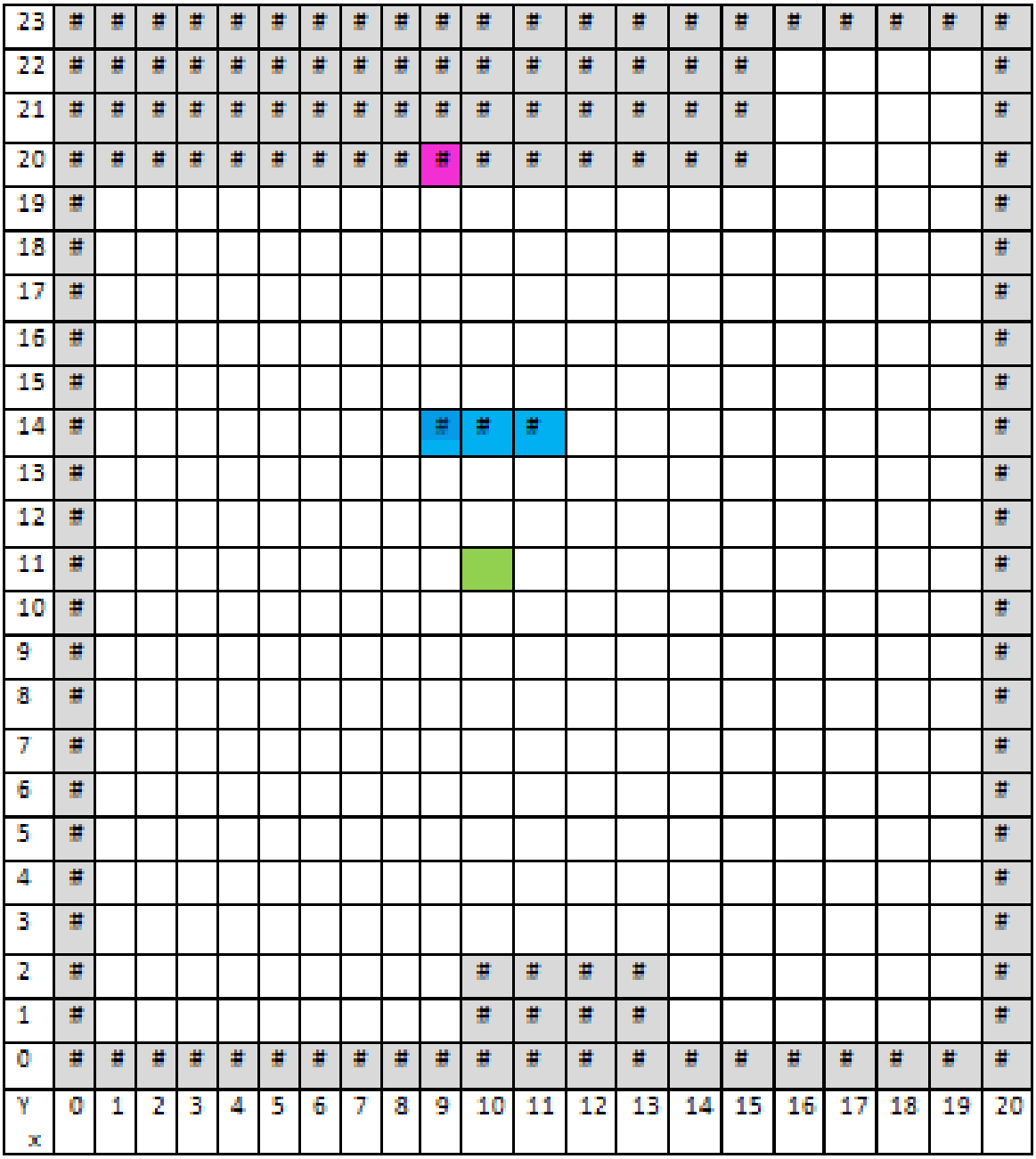
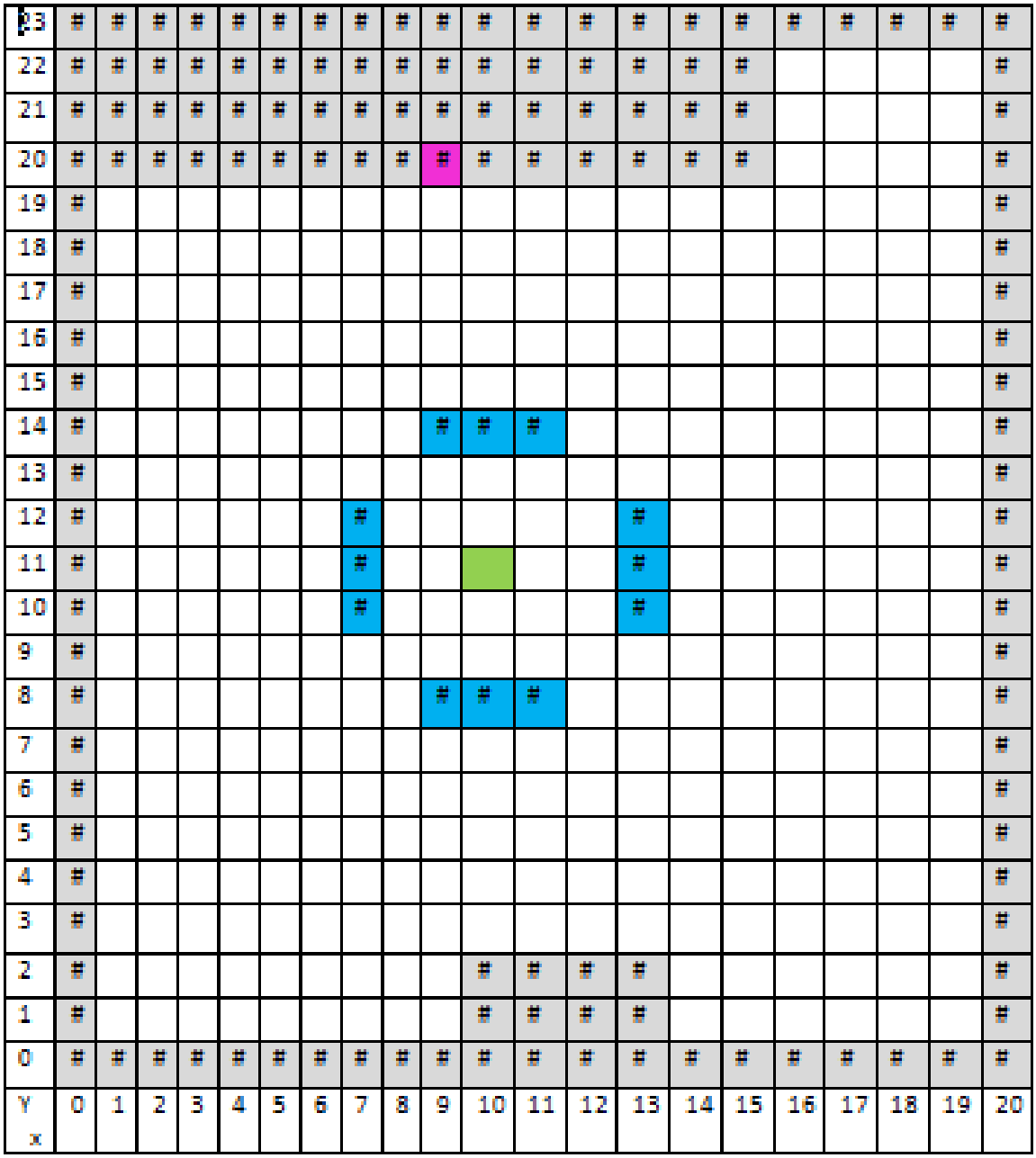


Figure 62 – Layout B



Y / x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
23	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
22	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#					#
21	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#					#
20	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#					#
19	#																				#
18	#																				#
17	#																				#
16	#																				#
15	#																				#
14	#									#	#	#									#
13	#																				#
12	#							#						#							#
11	#							#						#							#
10	#							#						#							#
9	#																				#
8	#									#	#	#									#
7	#																				#
6	#																				#
5	#																				#
4	#																				#
3	#																				#
2	#										#	#	#	#							#
1	#										#	#	#	#							#
0	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#

Figure 63 – Layout C

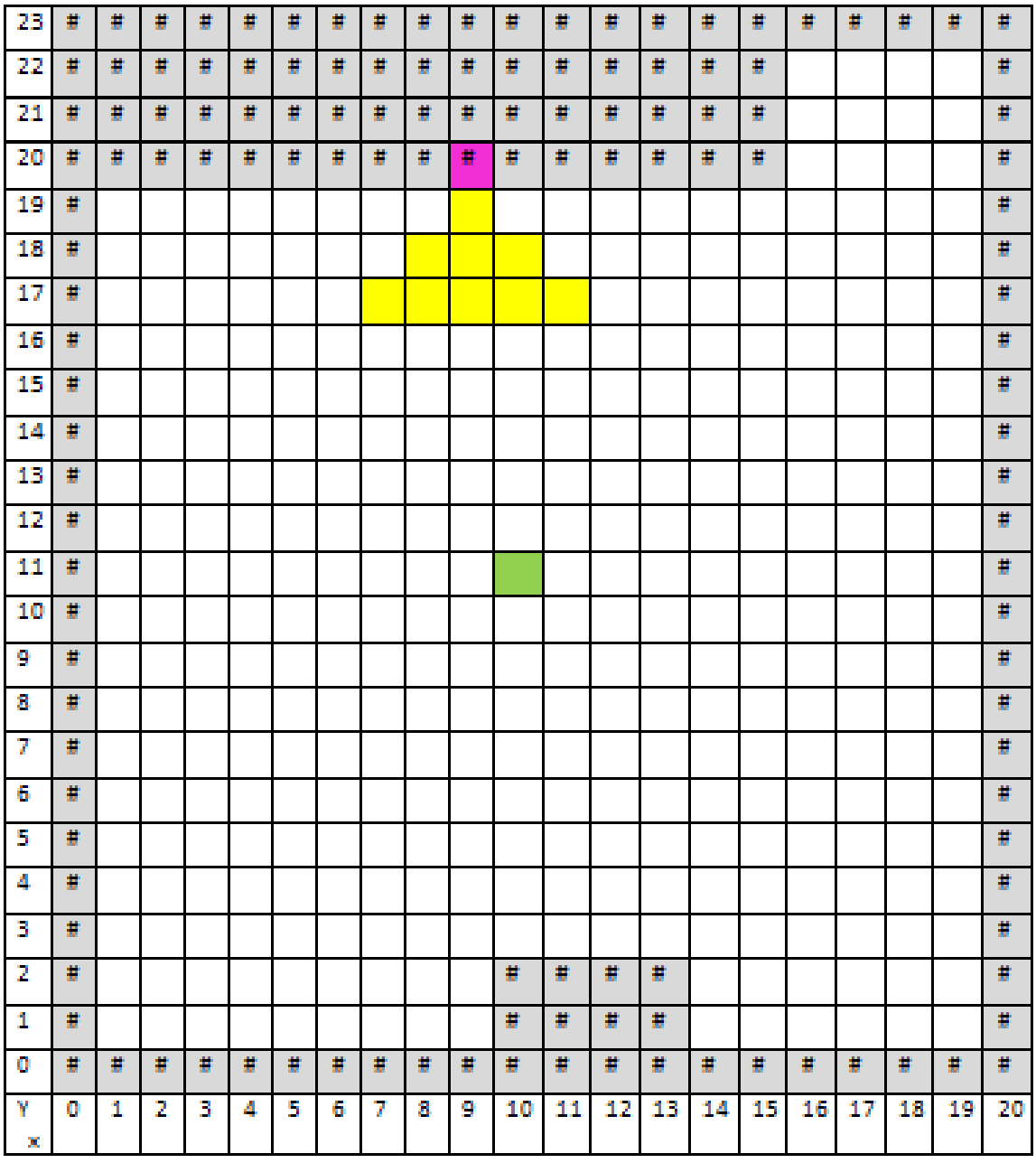


Figure 64 – Visibility Area

The above figure shows in yellow the area that the drone would need to be in to get a good match with the object. The drone could be facing any direction except away from the object to get a match in this yellow area.

4.2.3.5 Spiral

The spiral search pattern is the least advanced method of navigating a unknown environment. The robot moves outwards in a spiral direction until it either finds the object it is looking for, or comes across an obstacle. This method is not designed to deal with obstacles in the room and will assume it has reached the end of the search when it comes across one.

4.2.3.6 Wall Following

The wall following algorithm was the second most advanced and returned the best results. The robot heads straight ahead until it reaches a wall, and then continues to move sideways to the right while keeping the camera on the wall. At junctions the robot will rotate its orientation to face the new wall.

There are possible issues with this method of search, being that some wall configurations can put the robot in a loop around parts of the room, or with an alternate programming the robot could get caught looping around an object.

4.2.3.7 Zone

The zone algorithm was the hardest of them all. The robot took measurements from the 4 axis round it and assumed that the room was square and there were no obstacles. The room was them split into 4 and the robot proceeds to head to the centre of each zone and sun around to do a search. This proved difficult as the algorithm is designed for known search spaces, whereas this robot had to make do. If it came across an obstacle it would just head to the next zone.

To make this algorithm more effective it needs to learn the layout of a room and then map it into zones of about 3 meters by 3 meters to get an effective search. Ways of dealing with navigating around obstacles and learning new areas would also add to this.

4.2.3.8 Results

The following results table shows the results of testing with the three search pattern algorithms created.

Algorithm	Layout	Test Number	Obstacles Avoided?	Blocks travelled
Spiral	Figure 57 -A	1	Y	149
Spiral	Figure 58 -B	2	N	-
Spiral	Figure 59 -C	3	N	-
Wall	Figure 57 -A	4	Y	45
Wall	Figure 58 -B	5	Y	45
Wall	Figure 59 -C	6	Y	45
Zone	Figure 57 -A	7	Y	-
Zone	Figure 58 -B	8	Y	-
Zone	Figure 59 -C	9	Y	-

Figure 65 – Search Algorithm Results

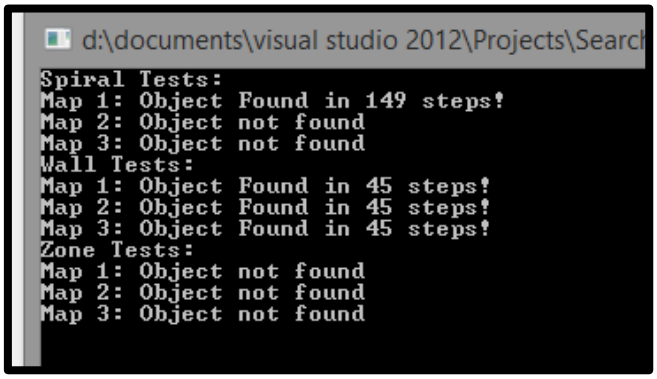


Figure 66- Search pattern tests

4.2.3.9 Conclusions

From the results table we can clearly see that the most effective method for finding the object for the set up that we have (object on wall), was the wall following algorithm. In second place was the spiral algorithm, working well in an empty room but talking a long time to travel to the edges, this algorithm got caught on any objects that where in its path.

Zone search was the worst of all the algorithms. A 4 zone grid wasn't small enough to deal with the size of the room and as such, searches where less accurate as they were trying to cover a bigger area.

If the object was not located on a wall then the wall following algorithm would be useless and the zone or spiral algorithms would have to be used. Spiral takes a long time to find its target and zone has issues dealing with odd shaped rooms unknown to it. While the wall following algorithm was best for this scenario, I would recommend a more advanced learning version of the zone algorithm o find an object located away from the walls.

5.0 Final Implementation

The final prototype implements the results of my research, combining the image processing module that I have programmed with the autonomous navigation module that was designed in the previous section.

5.1 Prototype Design

5.1.1 Hardware

From my secondary research I decided on the hardware that would be used. This was the AR Drone, a raspberry Pi, and 4 sonar sensors. Extra hardware includes resisters and ordinary circuit wire. The Pi will have a Wi-Fi card attached to allow it to talk to the drone and any other monitoring device.

5.1.2 Design

5.1.2.1 Build

The prototype will be built by attaching the Pi and sonar sensors to the AR Drone. This will be done with electrical tape to prevent the devices being permanently attached to the drone and preventing future alterations from being altered.

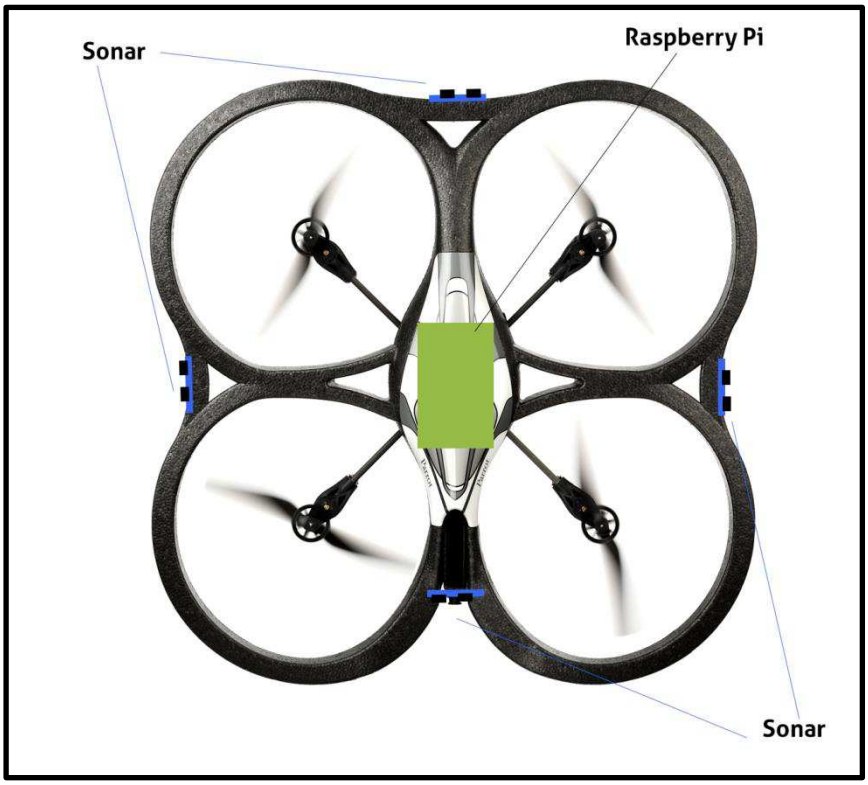


Figure 67 – AR Drone Build Schematic [38]

The schematic above shows the AR Drone, on this image is a diagram of where the components will be mounted to the indoor frame. The 4 sonar sensors will be placed on the 4 sides of the drone, with the Pi being mounted centrally.

5.1.2.2 Wiring Schematic

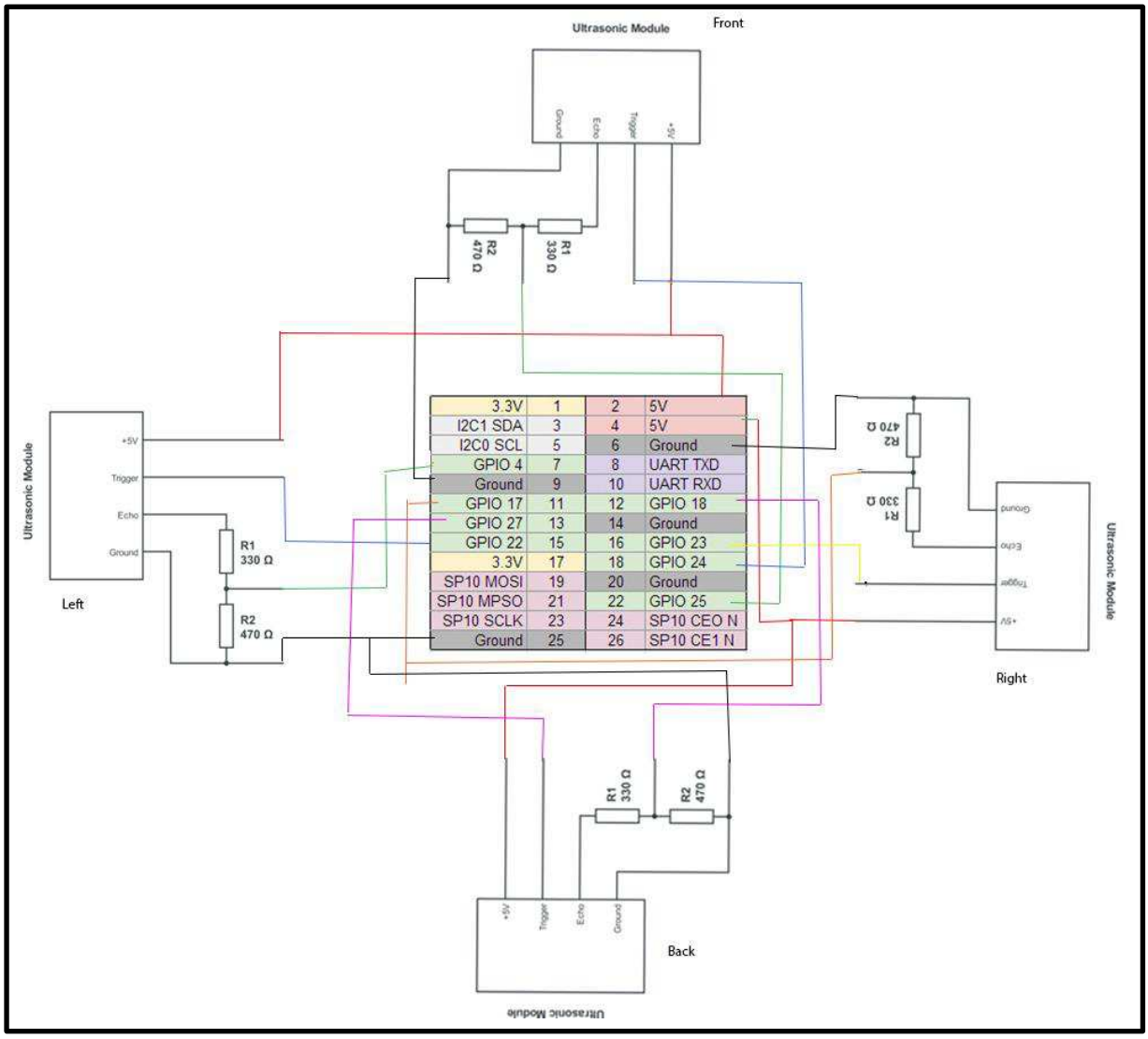


Figure 68 – Wiring Diagram

The above figure graphically represents the planned wiring plan for attaching the sonar sensors to the AR drone. Each ultrasonic module has 2 GPIO pins allocated to it, and are sharing a +5v and ground pin. When I wire the sonar sensors to the Pi, I will need to ensure I use shrink wrap and electrical tape due to the close proximity of the pins on the Pi.

5.1.3 Framework

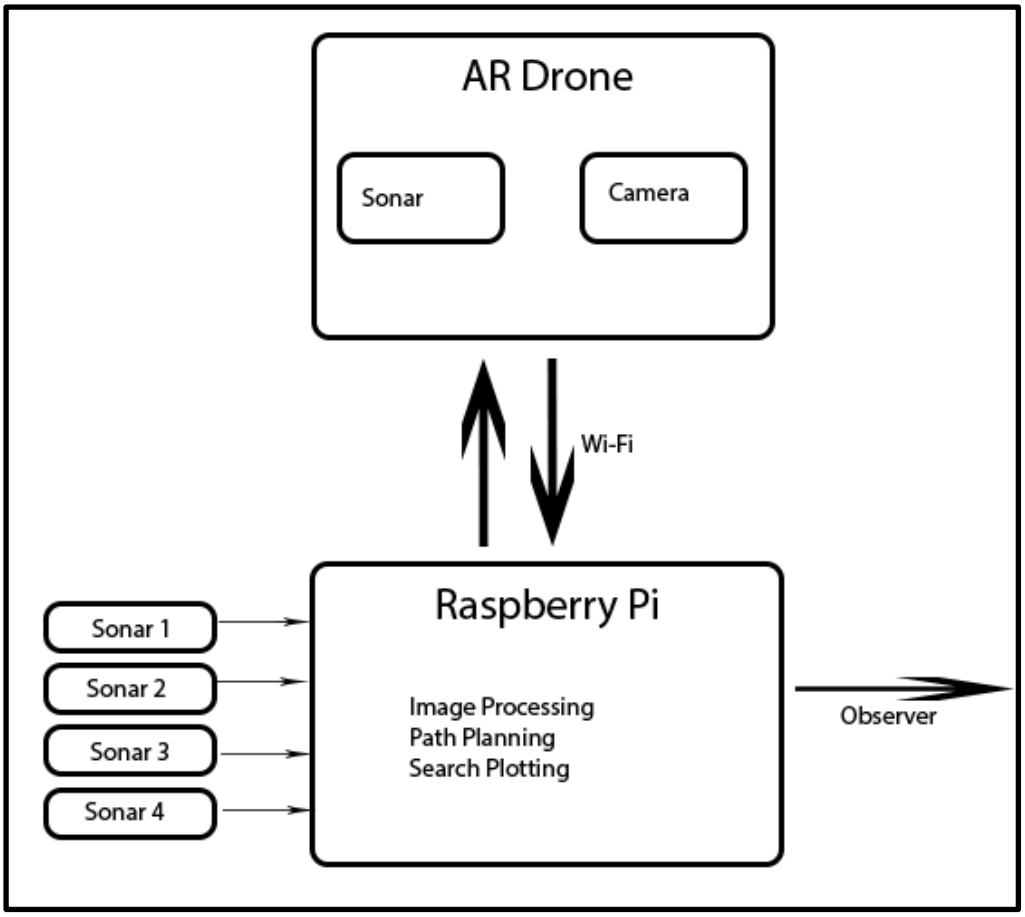


Figure 69 – System Architecture

The above figure details the system architecture for the project. At the bottom of the figure is the Raspberry Pi, with the attached sonar sensors. The Pi will manage all the "thinking" of the project, and will communicate with the AR Drone over a Wi-Fi connection. The AR Drone will communicate back with the Pi with images from the camera, and readings from the sonar sensor on the belly of the drone.

The following figure (Figure 70 – Program Flow) shows a graphical representation of the drones reasoning system. The drone takes off and initially searches for the object in its field of view. The drone then starts to perform a searching pattern and will also analyse video at the same time looking for the object in its field of view. When found the drone will land safely.

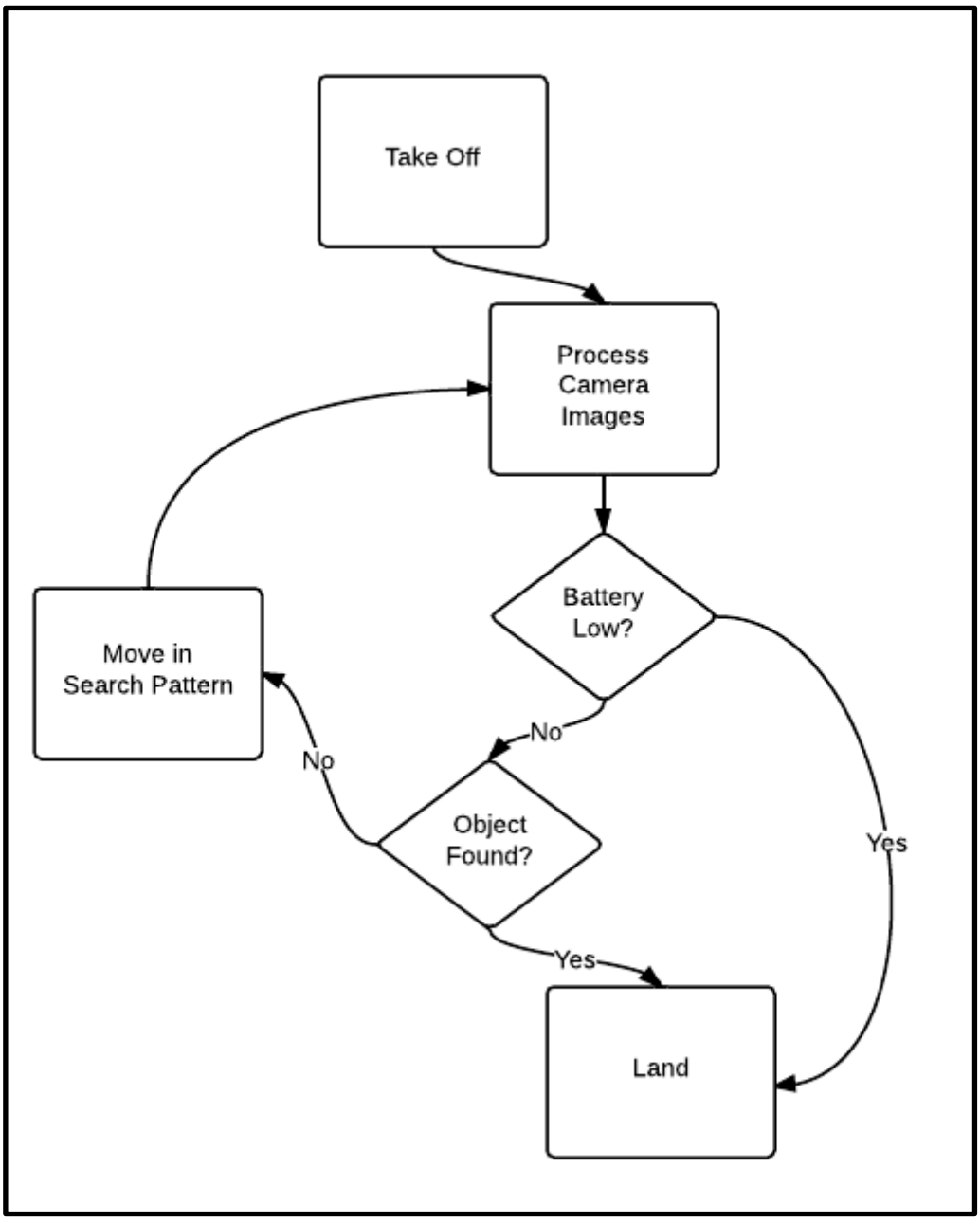


Figure 70 – Program Flow

These two framework diagrams lay out a high level overview of how the drone will operate in order to achieve its goals, and how the different parts of the system will operate with each other and work together.

5.2 Prototype Building

5.2.1 Building

Below are some pictures of the build process (Figure 71 – Drone Build and Circuitry, Figure 72 – Drone Build and Circuitry). I started with the schematics [27] for the sonar sensors and soldered wire to the ports with resisters, and then shrink wrapped the wire to keep it neat and prevent short circuits.

Figure 71 – Drone Build and Circuitry

The sensors where then mapped and soldered to the correct spare GPIO ports of the Raspberry Pi. Wire was then taped out of the way with electrical tape. Spare wire and the ground wire where bunched up above the board, to try and keep the weight central.

Figure 72 – Drone Build and Circuitry

5.2.2 Raspberry Pi Setup

I set up the Pi with a Raspberry Pi NOOBS build of Debian with a GUI to make it easier to program and also oversee what is going on with the drone while it is in flight.

To use the Raspberry Pi for the project I need a way of remotely activating the drone without connecting extra wires to the drone. The remote connection will also act as an emergency cut-out if it is needed. To do this I used a remote desktop connection which required installing “XRDP” [39].

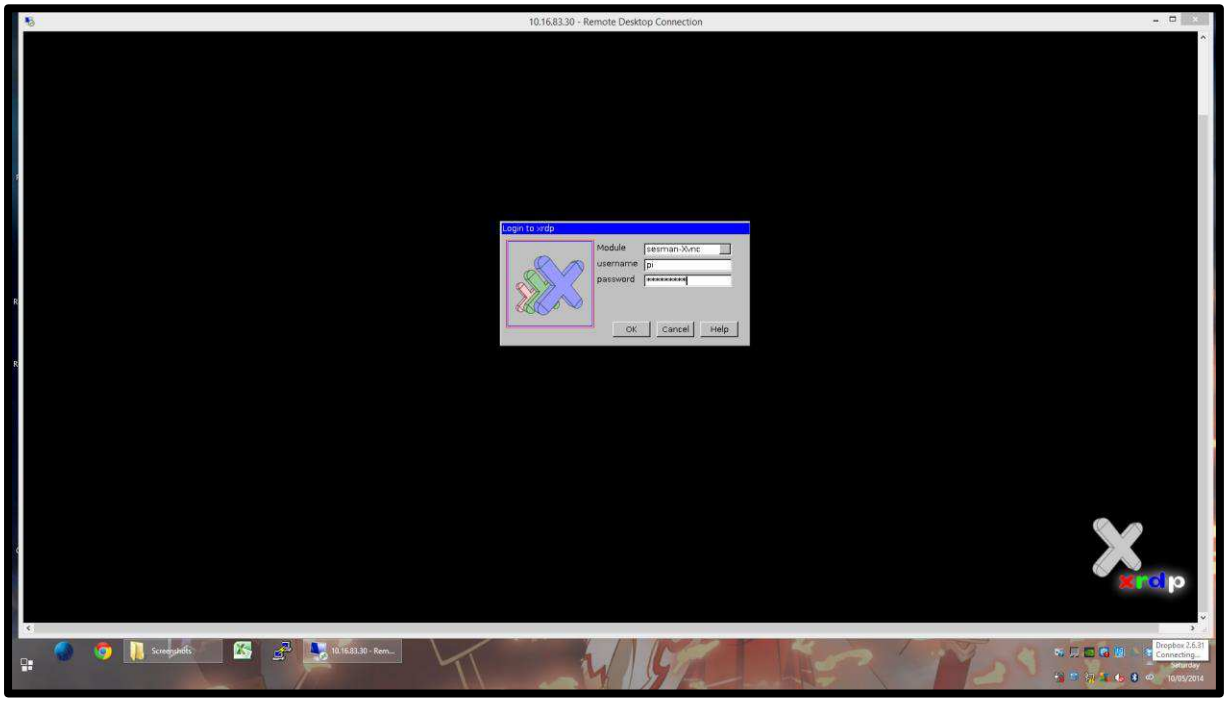


Figure 73 – Remote connection

5.3 Prototype Testing

5.3.1 Stable Flight (Pre-programming)

After building the prototype there was some testing that I undertook before using the platform for the testing of the navigation element of the project. These tests took the form of flight stability.

During the initial tests it was found that the drone was too heavy to take off with the sonar equipment and the raspberry pi attached. One way to fix this could be to increase the voltage output for the motors. This is not documented and left me to explore the configuration files on the AR drone.

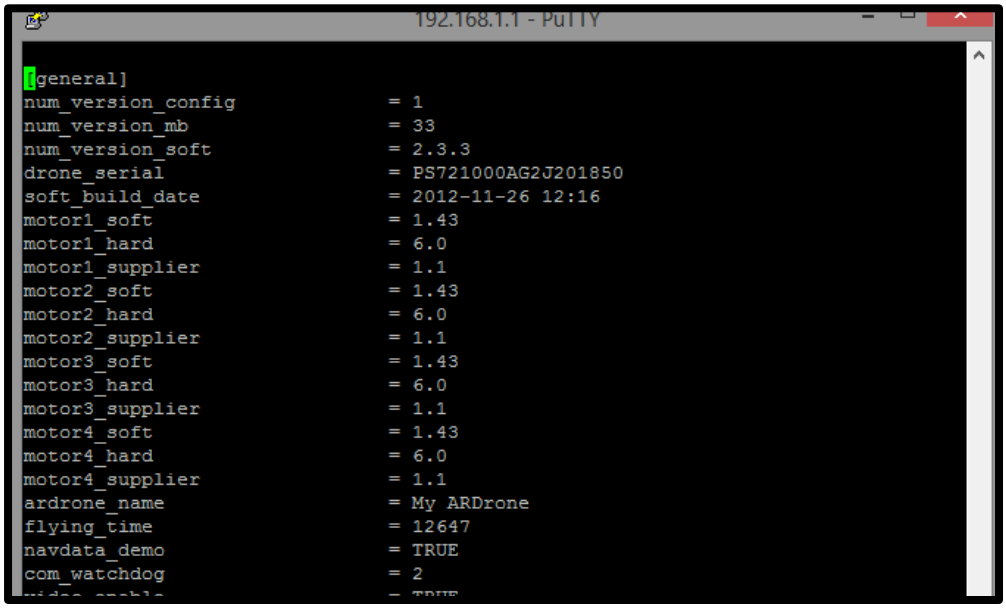


Figure 74 – Drone configuration file

Telnet to the drone was available on the IP address “192.168.1.1”. In the configuration file located at “data\conig.ini” there was data pertaining to the motors but unfortunately I eventually found these values to be the manufacturer, software version, and hardware version.

Because I could not change the voltage I next looked to reducing the weight of the drone. My first step was to change the hull from the indoor shell to the outdoor shell. This was successful in reducing the weight enough to allow the drone to take off.

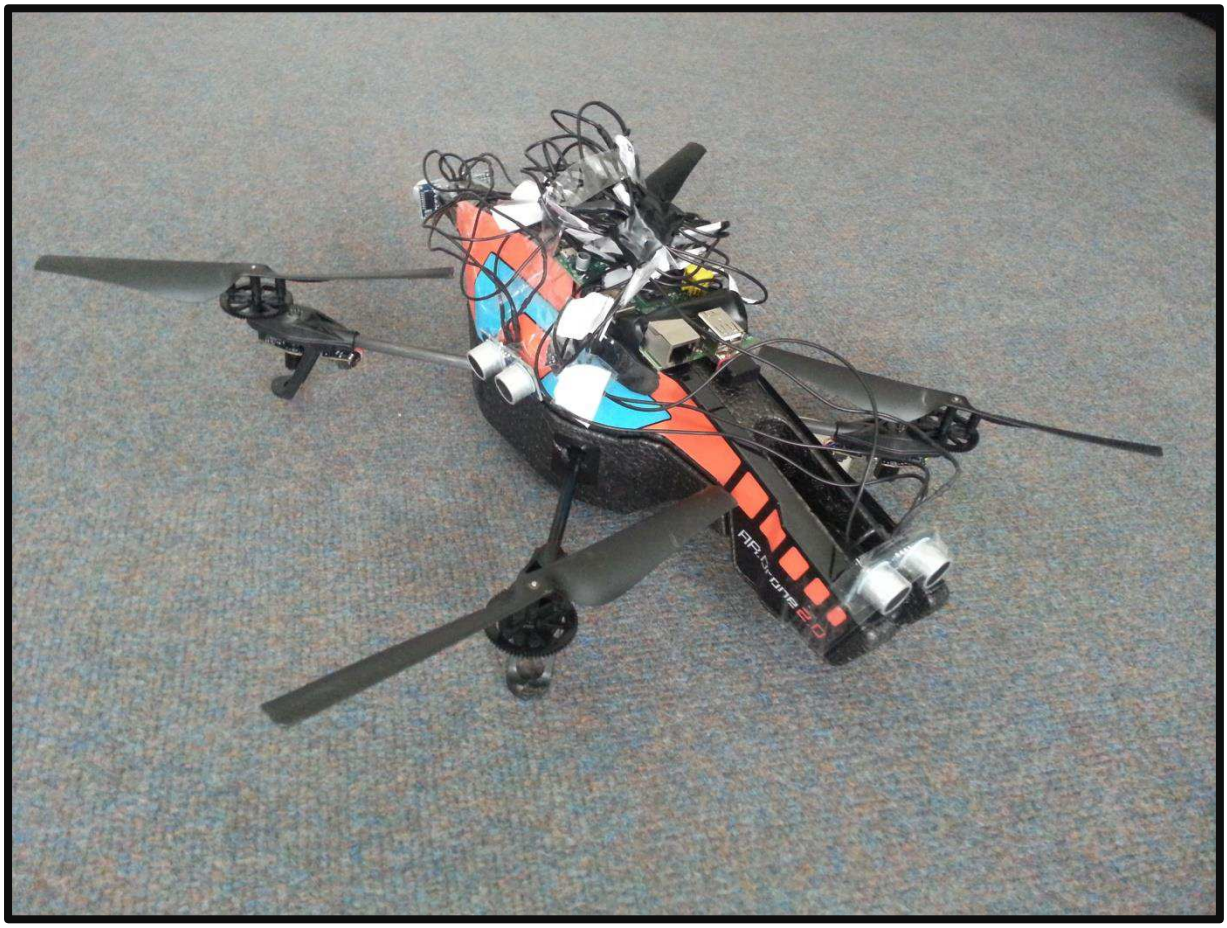


Figure 75 – Reconfigured Shell

The drone also required balancing. The distribution of weight across the drone was not optimal, so it caused the drone to drift and spin. To fix this I used small bits of blue tack to re-weight the drone so that it would be once again stable.

5.3.2 Sonar Sensors

Using python and some sample code from a tutorial for the sonar sensor I bought [27] I tested the sonar sensors out. Figure 76 shows the results of this test. I found that some surfaces are not good for use with these sensors as the readings back could erratically jump and that they were not as accurate as I had hopped. There has also been an issue connecting to and receiving feedback from the sonar sensor that measures distance behind the drone. Due to time constraints I have decided to manage without the rear sonar sensor when performing tests.

Code from the test is in the appndix (8.2.1 Sonar Sensor Test)

```
Distance Front : 260.1
Distance Right : 3248.4
Distance Left : 5.2
Distance Front : 331.2
Distance Right : 14.8
Distance Left : 807.2
Distance Front : 146.1
Distance Right : 3215.7
Distance Left : 3455.6
Distance Front : 148.2
Distance Right : 15.2
Distance Left : 5.2
Distance Front : 157.7
Distance Right : 3295.2
Distance Left : 3450.8
Distance Front : 156.8
Distance Right : 14.8
Distance Left : 5.3
Distance Front : 191.4
Distance Right : 3327.2
Distance Left : 3561.6
Distance Front : 146.2
Distance Right : 14.8
Distance Left : 5.6
Distance Front : 332.0
Distance Right : 3213.6
Distance Left : 3417.8
Distance Front : 115.8
Distance Right : 15.2
Distance Left : 5.2
Distance Front : 155.3
Distance Right : 3220.6
Distance Left : 3385.6
Distance Front : 146.9
Distance Right : 15.3
Distance Left : 5.2
^CTraceback (most recent call last):
  File "sonartest.py", line 82, in <module>
    stop4 = time.time()
KeyboardInterrupt
```

Figure 76 – Sonar Test

5.3.3 Raspberry Pi Issues

There were some issues with using the Raspberry pi and the AR Drone together. During compilation of the AR drone API for Linux the compile failed due to an error “bswap ip”. I found online that this issue is prevalent for many people attempting to use a raspberry Pi with an AR drone [40]. Unfortunately this issue seems to stem from the Pi’s use of an ARM processor and is not compatible with the AR Drone API. Fixing the issues related to this go beyond the scope of the project. To be able to continue with my research I can still capture images from the drone though the provided android app, which I can then test the visual algorithms with. As for navigation testing I can create a virtual test simulating the same environment to test which search pattern finds the object first.

6.0 Conclusion

6.1 Summary

I have found that both Template Matching and Pixel Classification are good methods for identifying a block of colour in an image; however a Pixel Classification method is more customisable and more decisive for the use with an autonomous robot.

The best search pattern algorithm I found was wall following, though this would only be effective for this scenario and further research would be required to produce a search algorithm that could learn its environment.

6.2 Evaluation

6.2.1 Project

The project suffered some setbacks during its lifecycle. Mainly being the issues of interfacing the Raspberry Pi with the AR Drone. To allow continuation of research and the project the topics where virtualised to represent best the real thing. Results from testing were sufficient to complete the projects requirements, although it is recommended to produce further research into learning algorithms for better results.

I found that it was possible to perform autonomous navigation and search in an indoor environment using an AR drone. At the same time I discovered that a Raspberry Pi is not the best interface for this task and that the small ultrasonic devices I used where not suitable for the rages that I was intending to use them for.

6.2.2 Project Management

Unforeseen circumstances saw some delays to my project after both project supervisors were taken ill for an extended period of time, causing me to become self-reliant in conducting my research. A revised project schedule is attached in the appendix for actual time taken.

At all times in the project I upheld the ethical issues discussed earlier in this report and endeavoured to conduct physical experiments safely in an environment without any observers except for the researcher.

The software development methodologies used worked well, with each sprint of the agile methodology being completed successfully with the iterative waterfall hybrid model. Code was designed, implemented, and tested simultaneously and iteratively.

I have learnt some valuable lessons from this project, such as providing some alternative methods of doing things if something goes wrong and doesn't work. I have also learnt that more time should be allocated to the practical research aspect to allow for unforeseen issues that may arise.

6.2.2.1 Viva

From my viva I received some feedback on my project on which I have enacted upon in this report. The transcript from the meeting is attached in the appendix (8.3 Viva). Outlined feedback for the project included, defining good/bad lighting levels, providing test layouts with graphics, improving upon the filtering process for the algorithms, improving flow charts, and a deeper look into the sensors (light, compass, gyroscope). In my report you can see all these items have been improved and extra sections included to cover the areas suggested in the viva feedback.

6.3 Recommendations

My recommendations for future continuation of the field would be to focus on developing a searching algorithm that can work well in unknown environments using machine learning to allow the drone to learn its surroundings and react much like a human would trying to perform the same task.

I would recommend attempting to use a different technology to work with the ultrasonic sensors than the Raspberry Pi due to compatibility issues and also speed issues. An app to allow commands to be sent to the drone would be appropriate as the project is about being mobile. The following section on future work outlines more possibilities for continuation of this field and project.

6.4 Future Work

There are many possibilities for the future continuation of this field and project. Listed below are some of my ideas for further development that would improve the effectiveness of autonomous search and navigation on quad-copters

6.4.1 Heat camera

I looked into a heat/IR camera during my research. At the time it was more complicated work beyond the scope of this project, and extra weight that would need to be carried on the drone. Adding this hardware to the drone would add an extra safety feature to the drone in preventing collisions with humans. It would also allow for the development of object tracking in parallel with the on-board camera. Humans being naturally warm would

be tracked easier and the drone could be configured to follow a human. This could also aid in search and rescue in identifying survivors.

6.4.2 Larger drone

A larger drone would benefit the project. The increased size would allow for more battery power for a longer flight time and for more sensors to be added without the worry of weight. An octa-copter (8 blades) could be a possibility for this, and would provide extra stability during flight.

6.4.3 Face recognition

Face recognition would add a level of recognition to the drone. You could ask the drone to locate a saved user that it already has a photo of, ideal for identifying individuals in a crowd, or for an autonomous message delivery system?

6.4.4 GPS

A GPS unit would be for the use of a drone in an outdoor environment. If permission was gained for an autonomous Ariel vehicle to be flown, then a GPS could be used to allow the drone to know its location and height, and also allow it to navigate great distances without using the sonar sensors.

6.4.5 Speakers

A small speaker could be added to the drone. While this doesn't affect the operation of the autonomous vehicle, it could be used to communicate with users. It could give audible warnings to people in areas that are unsafe to enter, or even deliver messages.

6.4.6 Object Detection

Object detection could be furthered. At current the drone is only recognising blocks of colour. Adding the ability to detect and identify objects adds more uses for the drone in a real word application. Identifying objects that may also be a danger and an obstruction to the drone's flight would also be a bonus.

6.4.7 Object Retrieval

Along with the last suggestion (Object Detection), the ability for the drone to pick up small objects would add to its practical implications. For example deliveries like Amazons Prime Air [5], or the retrieval and disposal of bombs.

6.4.8 Light Sensor

A light sensor is not an obvious choice for a future development. The attachment of a light sensor near the camera of the drone will allow the drone to measure what the lighting

conditions are; it can then make compensations in image enhancement to deal with high or low lighting.

6.4.9 Advanced Search Patterns

Beyond the scope of this dissertation there are more advanced methods of searching a room. I used very low level autonomous search pattern methods, but higher level algorithms that learn as they navigate around the solution space would be better suited to the task.

6.4.10 Mobile App

A mobile app would allow for more portable control of the project. You could give the drone commands on what to do from the app and even take pictures from the app for the drone to then locate. Commands given could range from “find this object”, “find me”, “find this person”, and “survey this room”

7.0 Bibliography

[1] "Code of Ethics - Drone Journalism," [Online]. Available: http://www.dronejournalism.org/code-of-ethics/.

[2] M. Abbott, "Amazon Drones Privacy - Truth Out," 4 January 2014. [Online]. Available: http://www.truth-out.org/news/item/21015-amazoncom-drones-raise-red-flags-regarding-privacy-rights. [Accessed 2014 05 2014].

[3] *The Air Navigation Order,* 2009.

[4] "ArduCopter," [Online]. Available: http://copter.ardupilot.com/.

[5] Amazon, "Amazon Prime Air," [Online]. Available: http://www.amazon.com/b?ie=UTF8&node=8037720011.

[6] S. L. Achal D Arvind, "Autonomous Navigation, object detection and retrieval in an Indoor Environment," 2012.

[7] "The Robot Operating System," [Online]. Available: http://wiki.ros.org/.

[8] Itseez, "Open Source Computer Vision," [Online]. Available: http://opencv.org/.

[9] J. J. Lugo and A. Zell, "Framework for Autonomous Onboard Navigationwith the AR.Drone," 2013.

[10] M. Saska, T. Krajńık, J. Faigl, V. Vońasek and L. Preucil, "Low cost MAV platform AR-drone in experimental verifications ofmethods for vision based autonomous navigation," in *International Conference on Intelligent Robots and Systems*, 2012.

[11] K. E. Wenzel, A. Masselli and A. Zell, "Visual Tracking and Following of a Quadrocopter by another Quadrocopter," in *International Conference on*, 2012.

[12] S. P. Soundararaj, A. K. Sujeeth and A. Saxena, "Autonomous Indoor Helicopter Flight using a Single Onboard Camera," in *J International Conference on*, 2009.

[13] Western University Canada, [Online]. Available: http://www.csd.uwo.ca/courses/CS4487a/Lectures/lec01_intro.pdf.

[14] Math Works, "Digital Image Processing," [Online]. Available: http://www.mathworks.co.uk/discovery/digital-image-processing.html.

[15] H. R. Myler and A. R. Weeks, The Pocket Handbook of Image Processing Algorithms, 1993.

[16] Y. Wang, "Image Filtering," [Online]. Available: http://eeweb.poly.edu/~yao/EE3414/image_filtering.pdf.

[17] P. David, "Reducing Noise in Photographs," [Online]. Available: http://petapixel.com/2013/05/29/a-look-at-reducing-noise-in-photographs-using-median-blending/.

[18] opencv, "Template Matching," [Online]. Available: http://docs.opencv.org/doc/tutorials/imgproc/histograms/template_matching/template_matching.html#template-matching.

[19] Society of Robots, "Computer Vision Algorithms," [Online]. Available: http://www.societyofrobots.com/programming_computer_vision_tutorial_pt3.shtml#image_correlation.

[20] SmartDraw, "Crime Scene Investigation Search Patterns," [Online]. Available: https://www.smartdraw.com/examples/view/crime+scene+investigation+search+patterns/.

[21] nakkaya, "Path Finding using Rapidly-Exploring Random Tree," [Online]. Available: http://nakkaya.com/2011/10/27/path-finding-using-rapidly-exploring-random-tree/.

[22] Sky Lab, "Engenius Lab wall follower," [Online]. Available: http://www.engeniuslab.com/embedded-lab/microcontrollers/avr/wall-follower-robot-by-using-atmega16-microcontroller/#tab-id-2.

[23] Parrot, "AR drone 2," [Online]. Available: http://ardrone2.parrot.com/.

[24] "Parrot AR Drone Forum," 2012. [Online]. Available: http://forum.parrot.com/ardrone/en/viewtopic.php?id=4545.

[25] A. Lappin, "Raspberry Pi with Python," [Online]. Available: http://www.bytecreation.com/blog/2013/10/13/raspberry-pi-ultrasonic-sensor-hc-sr04.

[26] Elec Freaks, "Ultrasonic Ranging Module HC - SR04," [Online]. Available: http://www.micropik.com/PDF/HCSR04.pdf.

[27] Matt, "Ultrasonic Distance Measurement Using Python," December 2012. [Online]. Available: http://www.raspberrypi-spy.co.uk/2012/12/ultrasonic-distance-measurement-using-python-part-1/.

[28] raspberrypi, "Pi NoIR," [Online]. Available: http://www.raspberrypi.org/tag/pi-noir/.

[29] adafruit, "IR distance sensor," [Online]. Available: http://www.adafruit.com/products/164.

[30] S. Piskorski, N. Brulez, P. Eline and F. D'Haeyer, AR.Drone Developer Guide, Parrot.

[31] NodeCopter.js, "Drone Pramming," 2012. [Online]. Available: http://nodecopter.com/.

[32] Itseez, "OpenCV," [Online]. Available: http://opencv.org/.

[33] Open Source Robotics Foundation, "Robot Operating System," [Online]. Available: http://www.ros.org/.

[34] Parrot, "Play Store - AR Free Flight," [Online]. Available: https://play.google.com/store/apps/details?id=com.parrot.freeflight.

[35] OpenCV, "Template Matching," [Online]. Available: //http://docs.opencv.org/doc/tutorials/imgproc/histograms/template_matching/template_matching.html.

[36] OpenCV, "Operations with images," [Online]. Available: http://docs.opencv.org/doc/user_guide/ug_mat.html.

[37] S. Fernando, "OpenCV Tutorial C++," [Online]. Available: http://opencv-srf.blogspot.co.uk/.

[38] "App Advice - Ar Done," [Online]. Available: http://appadvice.com/appnn/2010/06/parrot-ardrone-finally-official-release-date-price-app.

[39] A. Riley, "How to set up remote desktop for the Pi," [Online]. Available: http://www.raspberrypiblog.com/2012/10/how-to-setup-remote-desktop-from.html.

[40] ARDrone-Flyers, "SDK 2.0 and Raspberry Pi," [Online]. Available: http://www.ardrone-flyers.com/forum/viewtopic.php?f=23&t=3352.

8.0 Appendix

8.1 Meeting Records

Due to an unforeseen circumstance, Carey Pridgeon was no longer able to act as my primary supervisor for my project after November. Amanda Brooks as my second marker kindly stepped in to fill the roll. After January unforeseen circumstances meant that Amanda was unable to attend for several months. Due to this I have been self-disciplined in much of my work and any meetings needed where conducted over emails or instant messengers.

8.1.1 Carey Pridgeon

7th November 2013 – Project Proposal Writing Meeting

8.1.2 Amanda Brooks

3rd December 2013 – Report Structure, Lit review.

10th December 2013 – Literature review

7th January – Literature Review / Research Methodologies

21st January – Research Methodologies

28th January –Secondary Research

25 February –Secondary Research / Prototype Build

11th March – Secondary Research / Prototype Build

25th March – Primary Research

8th April – Primary Research

22nd April – Primary Research

7th May – Viva Presentation (1 hour approx.)

13th Mar – Hand In Date

8.2 Code

8.2.1 Sonar Sensor Test

```
#http://www.raspberrypi-spy.co.uk/2012/12/ultrasonic-distance-measurement-using-python-part-1/
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

GPIO_TRIGGER = 24 # front
GPIO_ECHO = 25
GPIO_TRIGGER2 = 23 # right
GPIO_ECHO2 = 17
GPIO_TRIGGER3 = 27 #back
GPIO_ECHO3 = 18
GPIO_TRIGGER4 = 22 #left
GPIO_ECHO4 = 4


print "Ultrasonic Measurement"

GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)
GPIO.setup(GPIO_TRIGGER2, GPIO.OUT)
GPIO.setup(GPIO_ECHO2, GPIO.IN)
GPIO.setup(GPIO_TRIGGER3, GPIO.OUT)
GPIO.setup(GPIO_ECHO3, GPIO.IN)
GPIO.setup(GPIO_TRIGGER4, GPIO.OUT)
GPIO.setup(GPIO_ECHO4, GPIO.IN)

GPIO.output(GPIO_TRIGGER, False)
GPIO.output(GPIO_TRIGGER2, False)
GPIO.output(GPIO_TRIGGER3, False)
GPIO.output(GPIO_TRIGGER4, False)


time.sleep(1.0)

while True:

    GPIO.output(GPIO_TRIGGER, True)
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)
    start = time.time()
    while GPIO.input(GPIO_ECHO)==0:
        start= time.time()

    while GPIO.input(GPIO_ECHO)==1:
        stop = time.time()

    elapsed = stop-start

    distance = elapsed * 34000

    distance = distance/2
```

```
    #----------------------------------------------
    GPIO.output(GPIO_TRIGGER2, True)
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER2, False)
    start2 = time.time()
    while GPIO.input(GPIO_ECHO2)==0:
        start2= time.time()

    while GPIO.input(GPIO_ECHO2)==1:
        stop2 = time.time()

    elapsed2 = stop2-start2

    distance2 = elapsed2 * 34000

    distance2 = distance2 / 2

    #----------------------------------------

    #----------------------------------------------
    GPIO.output(GPIO_TRIGGER4, True)
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER4, False)
    start4 = time.time()
    while GPIO.input(GPIO_ECHO4)==0:
        start4= time.time()

    while GPIO.input(GPIO_ECHO4)==1:
        stop4 = time.time()

    elapsed4 = stop4-start4

    distance4 = elapsed4 * 34000

    distance4 = distance4 / 2

    #----------------------------------------

    #----------------------------------------------


    #----------------------------------------

    print "Distance Front : %.1f" % distance

    print "Distance Right : %.1f" % distance2

    print "Distance Left : %.1f" % distance4

    #print "Distance Back : %.1f" % distance3

GPIO.cleanup()
```

8.2.2 Image Processing

8.2.2.1 Template matching

```
//http://docs.opencv.org/doc/tutorials/imgproc/histograms/template_matching/template_matching.html

#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>
#include <stdio.h>

using namespace std;
using namespace cv;

/// Global Variables
Mat img; Mat templ; Mat result;
char* image_window = "Source Image";
char* result_window = "Result window";

int match_method;
int max_Trackbar = 5;

/// Function Headers
void MatchingMethod( int, void* );

/** @function main */
int main( int argc, char** argv )
{
for(int i= 1; i<21 ;++i){


                    string filename="D:/desktop/imagesfortesting/";
                    filename += to_string(i);
                    filename += ".jpg";


                      /// Load image and template
                      img = imread( filename );
                      templ = imread( "D:/desktop/imagesfortesting/template.jpg");

                      if (img.empty())
                    {
                    cout << "Error : Image cannot be loaded..!!" << endl;
                     cin.get();
                    return -1;
                    }


                      /// Create windows
                      namedWindow( image_window, CV_WINDOW_AUTOSIZE );
                      namedWindow( result_window, CV_WINDOW_AUTOSIZE );

                      /// Create Trackbar
                      char* trackbar_label = "Method: \n 0: SQDIFF \n 1: SQDIFF NORMED \n 2: TM CCORR \n 3: TM CCORR NORMED \n 4: TM COEFF \n 5: TM COEFF NORMED";
                      createTrackbar( trackbar_label, image_window, &match_method, max_Trackbar, MatchingMethod );

                      MatchingMethod( 0, 0 );
```

```
                        waitKey(0);
        }
 return 0;
}

/**
 * @function MatchingMethod
 * @brief Trackbar callback
 */
void MatchingMethod( int, void* )
{
  /// Source image to display
  Mat img_display;
  img.copyTo( img_display );

  /// Create the result matrix
  int result_cols =  img.cols - templ.cols + 1;
  int result_rows = img.rows - templ.rows + 1;

  result.create( result_cols, result_rows, CV_32FC1 );

  /// Do the Matching and Normalize
  matchTemplate( img, templ, result, match_method );
  normalize( result, result, 0, 1, NORM_MINMAX, -1, Mat() );

  /// Localizing the best match with minMaxLoc
  double minVal; double maxVal; Point minLoc; Point maxLoc;
  Point matchLoc;

  minMaxLoc( result, &minVal, &maxVal, &minLoc, &maxLoc, Mat() );

  /// For SQDIFF and SQDIFF_NORMED, the best matches are lower values. For all the
other methods, the higher the better
  if( match_method  == CV_TM_SQDIFF || match_method == CV_TM_SQDIFF_NORMED )
    { matchLoc = minLoc; }
  else
    { matchLoc = maxLoc; }
  //cout<< result.at<uchar>(5, 5);
  /// Show me what you got
  rectangle( img_display, matchLoc, Point( matchLoc.x + templ.cols , matchLoc.y +
templ.rows ), Scalar::all(0), 2, 8, 0 );
  rectangle( result, matchLoc, Point( matchLoc.x + templ.cols , matchLoc.y +
templ.rows ), Scalar::all(0), 2, 8, 0 );
  cout<<endl<<matchLoc;

  imshow( image_window, img_display );
  imshow( result_window, result );

  return;
}
```

8.2.2.2 Pixel Classification

```
#include "opencv2/highgui/highgui.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main( int argc, const char** argv )
{

for(int i= 1; i<22;++i){

                        string filename="D:/desktop/imagesfortesting/";
                        filename += to_string(i);
                        filename += ".jpg";
                        //filename = "D:/desktop/testimg3.jpg";
                        Mat img = imread(filename, CV_LOAD_IMAGE_UNCHANGED);
                        //cout<<"filename: "<<filename;
                        if (img.empty())
                        {
                        cout << "Error : Image cannot be loaded..!!" << endl;
                         cin.get();
                        return -1;
                        }


                        int pinkpixels =0;
                        //cout<<endl<<"Image size : "<<img.cols<<"x"<<img.rows;

                        for(int x =0 ; x< img.cols;++x){
                               for(int y=0; y<img.rows;++y){

                                       Vec3b intensity = img.at<Vec3b>(y, x);
                                       float blue = intensity.val[0];
                                       float green = intensity.val[1];
                                       float red = intensity.val[2];

                                       //test 3 pink colour ranges

      if(((red<110)&&(red>90))&&((green>30)&&(green<40))&&((blue>55)&&(blue<75))){++p
inkpixels;}
                                       else
if(((red<140)&&(red>125))&&((green>55)&&(green<65))&&((blue>75)&&(blue<95))){++pinkpix
els;}
                                       else
if(((red<125)&&(red>105))&&((green>50)&&(green<70))&&((blue>75)&&(blue<95))){++pinkpix
els;}

                               }}
                        cout<<"Image: "<<i<<" -- Pink Pixels: "<<pinkpixels;
                        if(pinkpixels>200){cout<<"  -- Object: Detected";}else cout<<"  -
- Object: NOT Detected";
                        cout<<endl;
}
cin.get();


return 0;
}
```

8.2.3 Autonomous Navigation

```
#include <iostream>
#include <string>

using namespace std;

enum direction {up,down,Left,Right};

class map
{
	char grid[21][24]; //21x24 grid
	int dronex;
	int droney;
	direction dronedir;
	void load();
	int steps;
	int distanceUp();
	int distanceDown();
	int distanceLeft();
	int distanceRight();
public:
	map(){load();steps =0;	dronex=10;	droney=11;dronedir=down;}
	void getdronepos(int &x, int &y){x=dronex; y=droney;}
	int checkwall(int x, int y){if(grid[x][y]=='#')return 1; else return 0;}
	void movedrone(direction dir){if(dir == up){droney=droney+1;}else if(dir ==
down){droney=droney-1;}else if(dir == Right){dronex=dronex+1;}else if(dir ==
Left){dronex=dronex-1;}++steps;}//cout<<"\nDrone Position X:"<<dronex<<"
Y:"<<droney;++steps;}
	void movedrone(){if(dronedir == up){droney=droney+1;;}else if(dronedir ==
down){droney=droney-1;}else if(dronedir == Right){dronex=dronex+1;}else if(dronedir ==
Left){dronex=dronex-1;}++steps;}//else cout<<"match fail"; cout<<"\nDrone Position
X:"<<dronex<<" Y:"<<droney;++steps;}
	direction getDir(){return dronedir;}
	void changeDir(direction dir){dronedir = dir;}
	int getsteps(){return steps;}
	int objectvisable();
	void print();

	int LeftSonar() //simulates sonar with consideration for drone direction
	{
		if(dronedir==up) return distanceLeft();
		if(dronedir==down) return distanceRight();
		if(dronedir==Left) return distanceDown();
		if(dronedir==Right) return distanceUp();
	}

	int RightSonar() //simulates sonar with consideration for drone direction
	{
		if(dronedir==up) return distanceRight();
		if(dronedir==down) return distanceLeft();
		if(dronedir==Left) return distanceUp();
		if(dronedir==Right) return distanceDown();
	}


	int FrontSonar() //simulates sonar with consideration for drone direction
	{
		if(dronedir==up) return distanceUp();
```

```
		if(dronedir==down) return distanceDown();
		if(dronedir==Left) return distanceLeft();
		if(dronedir==Right) return distanceRight();
	}

	int BackSonar() //simulates sonar with consideration for drone direction
	{
		if(dronedir==up) return distanceDown();
		if(dronedir==down) return distanceUp();
		if(dronedir==Left) return distanceRight();
		if(dronedir==Right) return distanceLeft();
	}


	void loadmap1(){}//do nothing, no extras to add to grid
	void loadmap2(){grid[9][14]='#';grid[10][14]='#';grid[11][14]='#'; }
	void loadmap3();

};

class spiralsearch
{
	int stepLength;
	int stepcounter;
	map mymap;

public:
	spiralsearch(map testmap){stepLength=1; stepcounter=0;mymap=testmap;}
	int getsteps(){return mymap.getsteps();}
	void loadmap2(){mymap.loadmap2();}
	void loadmap3(){mymap.loadmap3();}
	int run()
	{
		while(true)
		{
			int tempsteplen = stepLength;
			for(int i=1; i<=tempsteplen; ++i)
			{
				if(mymap.objectvisable()==1) return 1;
				//cout<<"Sonar("<<mymap.FrontSonar()<<")
Direction("<<mymap.getDir()<<")";
				if(mymap.FrontSonar()==0) return 0;

				mymap.movedrone();
				if(i==stepLength)
				{//change direction
					if(mymap.getDir() == up) mymap.changeDir(Left);
					else if(mymap.getDir() == Left)
mymap.changeDir(down);
						else if(mymap.getDir() == down)
mymap.changeDir(Right);
							else if(mymap.getDir() == Right)
mymap.changeDir(up);
				}


				//cin.get();
			}
```

```
                        if(stepcounter==1){stepcounter=0;++stepLength;}else
++stepcounter;
                }
        }

};

class zonesearch
{
        map mymap;
        int width;
        int height;

        int xpos, ypos;
        int zone1[2];
        int zone2[2];
        int zone3[2];
        int zone4[2];

public:
        zonesearch(map testmap){mymap=testmap;}
        int getsteps(){return mymap.getsteps();}
        void loadmap2(){mymap.loadmap2();}
        void loadmap3(){mymap.loadmap3();}
        int run()
        {
                if(mymap.objectvisable()==1) return 1;
                //split area into 4 zones
                width = mymap.LeftSonar()+mymap.RightSonar()+1;
                height = mymap.FrontSonar() + mymap.BackSonar()+1;
                mymap.changeDir(up);
                xpos= mymap.LeftSonar()+1;
                ypos =  mymap.BackSonar()+1;

                zone1[0]=width/4; //x
                zone1[1]=(height/4)*3+1; //y

                zone2[0]=(width/4)*3;//x
                zone2[1]=(height/4)*3+1; //y

                zone3[0]=width/4;//x
                zone3[1]=height/4+1;//y

                zone4[0]= (width/4)*3;//x
                zone4[1]= height/4+1;//y
                //cout<<"Xpos: "<<xpos<<" Ypos:"<<ypos;
                //travel to zone 1
                bool clear = true;
                while(clear)
                {
                        for(int i =0; i<(zone1[1]-ypos);++i)
                        {
                                if(mymap.FrontSonar()==0) clear=false;
                                else
                                {
                                        mymap.movedrone(); ++ypos;
                                        if(mymap.objectvisable()==1) return 1;

                                        int x,y;
```

```
                        mymap.getdronepos(x,y);
                        //cout<<"\n X:"<<x<<" Y:"<<y;
                        //cin.get();
                }
        }
        mymap.changeDir(Left);
        for(int i =0; i<=(xpos-zone1[0]);++i)
        {
                if(mymap.FrontSonar()==0) clear=false;
                else
                {
                        mymap.movedrone();--xpos;
                        if(mymap.objectvisable()==1) return 1;

                        int x,y;
                        mymap.getdronepos(x,y);
                        //cout<<"\n X:"<<x<<" Y:"<<y;
                        //cin.get();
                }
        }
        break;
}
mymap.changeDir(Left);
if(mymap.objectvisable()==1) return 1;
mymap.changeDir(up);
if(mymap.objectvisable()==1) return 1;
mymap.changeDir(down);
if(mymap.objectvisable()==1) return 1;
mymap.changeDir(Right);
if(mymap.objectvisable()==1) return 1;
//cout<<"Xpos: "<<xpos<<" Ypos:"<<ypos;
//travel to zone 2
clear = true;
while(clear)
{
        mymap.changeDir(Right);
        for(int i =0; i<=(zone2[0]-xpos);++i)
        {
                if(mymap.FrontSonar()==0) clear=false;
                else
                {
                        mymap.movedrone(); ++xpos;
                        if(mymap.objectvisable()==1) return 1;

                        int x,y;
                        mymap.getdronepos(x,y);
                        //cout<<"\n X:"<<x<<" Y:"<<y;
                        //cin.get();
                }
        }
        break;
}
mymap.changeDir(Left);
if(mymap.objectvisable()==1) return 1;
mymap.changeDir(up);
if(mymap.objectvisable()==1) return 1;
mymap.changeDir(down);
if(mymap.objectvisable()==1) return 1;
mymap.changeDir(Right);
```

```
if(mymap.objectvisable()==1) return 1;
//cout<<"Xpos: "<<xpos<<" Ypos:"<<ypos;

//travel to zone 4
clear = true;
while(clear)
{
	mymap.changeDir(down);
	for(int i =0; i<=(ypos-zone4[1]);++i)
	{
		if(mymap.FrontSonar()==0) clear=false;
		else
		{
			mymap.movedrone(); --ypos;
			if(mymap.objectvisable()==1) return 1;

			int x,y;
			mymap.getdronepos(x,y);
			//cout<<"\n X:"<<x<<" Y:"<<y;
			//cin.get();
		}
	}
	break;
}
mymap.changeDir(Left);
if(mymap.objectvisable()==1) return 1;
mymap.changeDir(up);
if(mymap.objectvisable()==1) return 1;
mymap.changeDir(down);
if(mymap.objectvisable()==1) return 1;
mymap.changeDir(Right);
if(mymap.objectvisable()==1) return 1;
//cout<<"Xpos: "<<xpos<<" Ypos:"<<ypos;

//travel to zone 3
clear = true;
while(clear)
{
	mymap.changeDir(Left);

	for(int i =0; i<=(xpos-zone3[0]);++i)
	{
		if(mymap.FrontSonar()==0) clear=false;
		else
		{
			mymap.movedrone(); --xpos;
			if(mymap.objectvisable()==1) return 1;

			int x,y;
			mymap.getdronepos(x,y);
			//cout<<"\n X:"<<x<<" Y:"<<y;
			//cin.get();
		}
	}
	break;
}
mymap.changeDir(Left);
if(mymap.objectvisable()==1) return 1;
mymap.changeDir(up);
```

```
		if(mymap.objectvisable()==1) return 1;
		mymap.changeDir(down);
		if(mymap.objectvisable()==1) return 1;
		mymap.changeDir(Right);
		if(mymap.objectvisable()==1) return 1;

		return 0;
	}

};

class wallsearch
{
	map mymap;
	int loopx, loopy;
public:
	wallsearch(map testmap){mymap=testmap;}
	int getsteps(){return mymap.getsteps();}
	void loadmap2(){mymap.loadmap2();}
	void loadmap3(){mymap.loadmap3();}
	int run(){
		int left, right, front, back;
		left=mymap.LeftSonar();
		right=mymap.RightSonar();
		front=mymap.FrontSonar();
		back=mymap.BackSonar();

		while(mymap.FrontSonar()!=1)//find wall
		{
			if(mymap.objectvisable()==1) return 1;
			mymap.movedrone();
			int x,y;
			mymap.getdronepos(x,y);
			loopx=x;
			loopy-y;
			///cout<<"\n X:"<<x<<" Y:"<<y;
		}

		//move clockwise
		while(true)
		{
			if(mymap.objectvisable()==1) return 1;

			if(mymap.FrontSonar()!=1)//find wall
			{
				mymap.movedrone();
			}
			else if(mymap.RightSonar()!=1)//move along wall to junction
			{
				direction drone;
				drone = mymap.getDir();
				if(drone==down)mymap.movedrone(Left);
				else if(drone==up)mymap.movedrone(Right);
				else if(drone==Left)mymap.movedrone(up);
				else if(drone==Right)mymap.movedrone(down);
			}
			else // at junction turn
			{
				direction drone;
```

```
                        drone = mymap.getDir();
                        if(drone==down)mymap.changeDir(Left);
                        else if(drone==up)mymap.changeDir(Right);
                        else if(drone==Left)mymap.changeDir(up);
                        else if(drone==Right)mymap.changeDir(down);
                }


                        int x,y;
                        mymap.getdronepos(x,y);
                        if((loopx==x)&&(loopy==y)) return 0; //stuck in
loop - not found

                        //cout<<"\n X:"<<x<<" Y:"<<y;
                        //cin.get();
        }

        return 0;
    }

};

//------------------------------------------------------------------------------
int main()
{
    cout<<"Spiral Tests: "<<endl;
    map spiralmap1; cout<<"Map 1: ";
    spiralsearch spiral1(spiralmap1);
    if(spiral1.run() == 1) cout<<"Object Found in "<<spiral1.getsteps()<<"
steps!"<<endl; else cout<<"Object not found"<<endl;
    map spiralmap2; cout<<"Map 2: ";
    spiralsearch spiral2(spiralmap2);
    spiral2.loadmap2();
    if(spiral2.run() == 1) cout<<"Object Found in "<<spiral2.getsteps()<<"
steps!"<<endl; else cout<<"Object not found"<<endl;
    map spiralmap3; cout<<"Map 3: ";
    spiralsearch spiral3(spiralmap3);
    spiral3.loadmap3();
    if(spiral3.run() == 1) cout<<"Object Found in "<<spiral3.getsteps()<<"
steps!"<<endl; else cout<<"Object not found"<<endl;

    cout<<"Wall Tests: "<<endl;
    map wallmap1; cout<<"Map 1: ";
    wallsearch wall1(wallmap1);
    if(wall1.run() == 1) cout<<"Object Found in "<<wall1.getsteps()<<"
steps!"<<endl; else cout<<"Object not found"<<endl;
    map wallmap2; cout<<"Map 2: ";
    wallsearch wall2(wallmap2);
    wall2.loadmap2();
    if(wall2.run() == 1) cout<<"Object Found in "<<wall2.getsteps()<<"
steps!"<<endl; else cout<<"Object not found"<<endl;
    map wallmap3; cout<<"Map 3: ";
    wallsearch wall3(wallmap3);
    wall3.loadmap3();
    if(wall3.run() == 1) cout<<"Object Found in "<<wall3.getsteps()<<"
steps!"<<endl; else cout<<"Object not found"<<endl;

    cout<<"Zone Tests: "<<endl;
    map zonemap1; cout<<"Map 1: ";
```

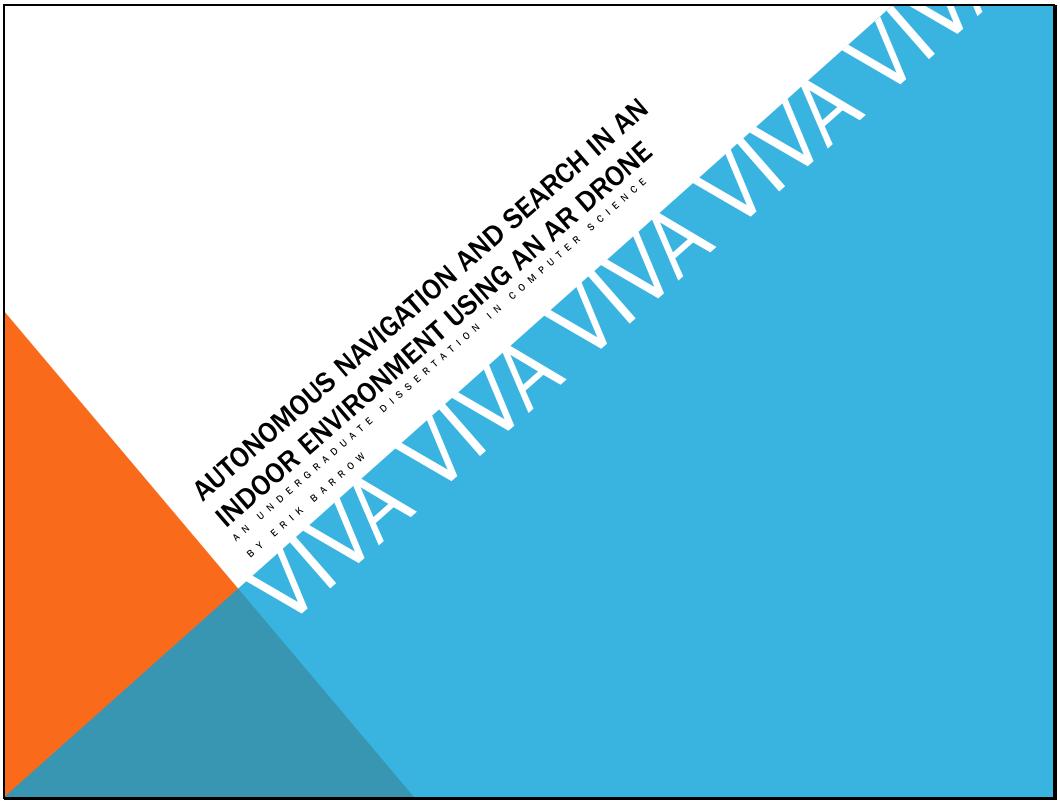
```
	zonesearch zone1(zonemap1);
	if(zone1.run() == 1) cout<<"Object Found in "<<zone1.getsteps()<<"
steps!"<<endl; else cout<<"Object not found"<<endl;
	map zonemap2; cout<<"Map 2: ";
	zonesearch zone2(zonemap2);
	zone2.loadmap2();
	if(zone2.run() == 1) cout<<"Object Found in "<<zone2.getsteps()<<"
steps!"<<endl; else cout<<"Object not found"<<endl;
	map zonemap3; cout<<"Map 3: ";
	zonesearch zone3(zonemap3);
	zone3.loadmap3();
	if(zone3.run() == 1) cout<<"Object Found in "<<zone3.getsteps()<<"
steps!"<<endl; else cout<<"Object not found"<<endl;

	cin.get();
	return 0;
}

//--------------------------------------------------------------------------------

void map::load()
{

	//cout<<"x:"<<dronex<<" y:"<<droney;

	for(int i=0; i<21; ++i)
	{
		for( int j =0; j<24;++j)
		{
			grid[i][j]=' ';
		}
	}

	for(int i=0; i<21; ++i)
	{
		grid[i][0]='#';
		grid[i][23]='#';
	}
	for(int i=0; i<24; ++i)
	{
		grid[0][i]='#';
		grid[20][i]='#';
	}
	for(int i=10; i<14; ++i)
	{
		grid[i][1]='#';
		grid[i][2]='#';
	}
	for(int i=1; i<16; ++i)
	{
		grid[i][22]='#';
		grid[i][21]='#';
		grid[i][20]='#';
	}


}

void map::loadmap3()
```

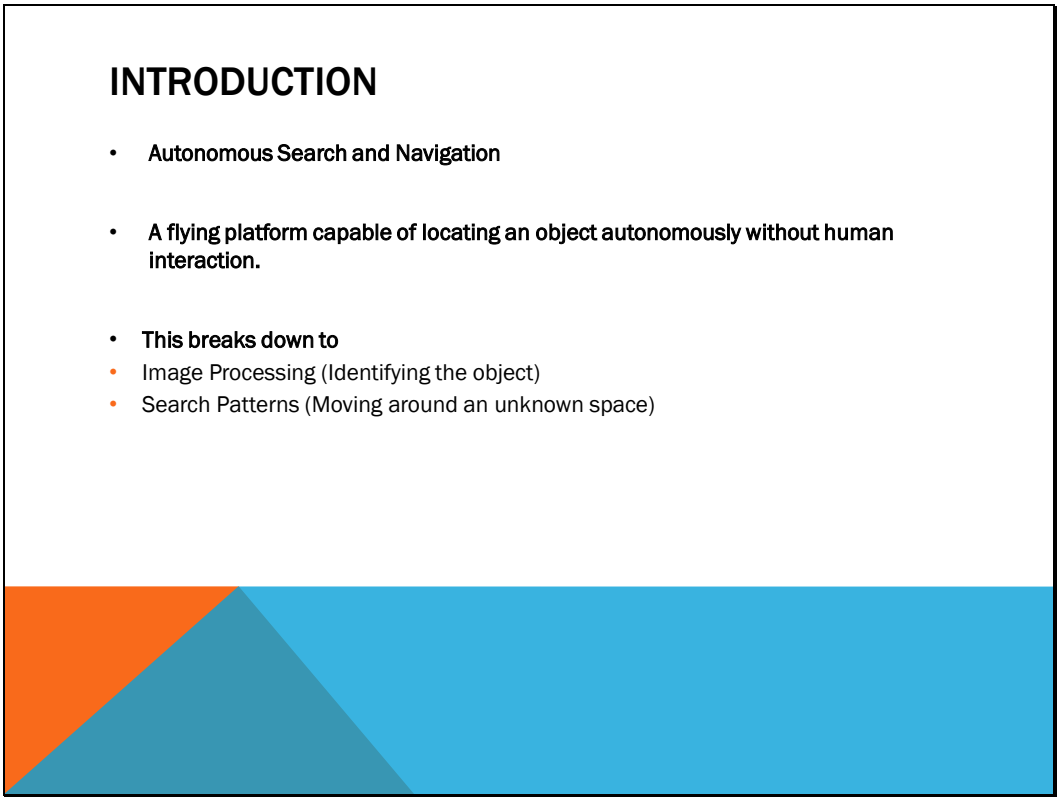
```
{
	grid[9][14]='#';grid[10][14]='#';grid[11][14]='#';
	grid[9][8]='#';grid[10][8]='#';grid[11][8]='#';
	grid[7][10]='#';grid[7][11]='#';grid[7][12]='#';
	grid[13][10]='#';grid[13][11]='#';grid[13][12]='#';

}

int map::objectvisable()
	{
		if (dronedir !=down)
		{
			if(dronex == 9 && droney == 19)return 1;
			if(dronex == 8 && droney == 18)return 1;
			if(dronex == 9 && droney == 18)return 1;
			if(dronex == 10 && droney == 18)return 1;
			if(dronex == 7 && droney == 17)return 1;
			if(dronex == 8 && droney == 17)return 1;
			if(dronex == 9 && droney == 17)return 1;
			if(dronex == 10 && droney == 17)return 1;
			if(dronex == 11 && droney == 17)return 1;

		}
		return 0;

}

int map::distanceUp()
{
	int dist=0;
	//int x,y;
	//getdronepos(x,y);
	for(int i = droney; i<24; ++i)
	{
		if(checkwall(dronex,i)==0)++dist;
		else
			return dist;
	}
	return dist;

}

int map::distanceDown()
{
	int dist=0;
	int x,y;
	getdronepos(x,y);
	for(int i = droney; i>0; --i)
	{
		if(checkwall(dronex,i)==0)++dist;
		else
		return dist;
	}
	return dist;

}

int map::distanceRight()
{
```

```
	int dist=0;
	int x,y;
	getdronepos(x,y);
	for(int i = dronex; i<21; ++i)
	{
		if(checkwall(i,droney)==0)++dist;
		else
			return dist;
	}
	return dist;

}

int map::distanceLeft()
{
	int dist=0;

	for(int i = dronex; i>0; --i)
	{
		if(checkwall(i,droney)==0)++dist;
		else
			return dist;
	}
	return dist;

}

void map::print()
{
	for(int j=23; j>=0; --j)
	{
		for(int i=0; i<21; ++i)
		{
			cout<<grid[i][j];
		}
		cout<<endl;
	}
}
```

8.3 Viva

8.3.1 Presentation





CONCEPT VIDEO

LIT REVIEW

ArduCopter

- Automated not autonomous
- Software
- Waypoint tracking

Amazon Prime Air

- Autonomous Flight?
- Object delivery

Autonomous Navigation, object detection, and retrieval in an indoor environment

- Autonomous
- Searches for and retrieves object
- Uses Arduino
- External path planning (Cluster of machines)

Framework for autonomous on-board navigation with the AR.Drone

- Uses as few sensors as possible
- Distance travelled estimated using gyroscope

LIT REVIEW

Low cost MAV platform AR-drone in experimental verifications of methods for vision based autonomous navigation

- Learning Algorithm
- Uses Camera
- Flown once to teach system

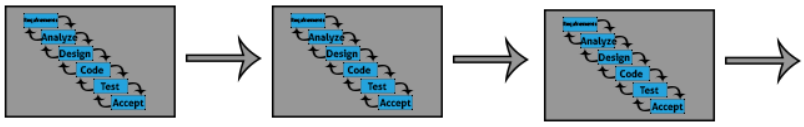
Visual tracking and following of a quadrocopter by another quadrocopter

- Smaller Quad-Copters following a larger Quad
- Uses IR Beacon to simplify tracking

Autonomous indoor helicopter flight using a single on board camera

- Helicopter not Quad
- Camera Based navigation
- Processed off device (computers)

METHODOLOGY



My software methodology For the algorithm testing stage

- Agile Outer Shell
- Inner Shell Waterfall/Iterative Hybrid
- Inner shell Represents each test

METHODOLOGY

Initial Algorithms Selection for Testing

Algorithm	Advantages	Disadvantages	Comments

Visual Algorithm Test Table

Algorithm	Test Number	Static?	Distance From Object	Lighting Level (Good/Poor)	Pass / Fail

Navigation Algorithm Test Table

Algorithm	Test Number	Obstacles Avoided?	Time Taken to Find Object

ALGORITHMS - VISUAL

Pre -Processing

De-Blurring

- Caused by camera / object movement
- Image essentially shifted
- Realigns image

Noise Reduction

- Dust
- Static
- Pixel Averaging

Sharpening

- Takes Edges in an image
- Adds detail to edges

ALGORITHMS - VISUAL

Template Matching

- Takes 2 images as input
- Smaller template and larger image to compare
- Template is slid around the large image to search for a match

Pixel Classification

- Colours are converted into object classes
- Pixel by pixel
- Pixel put into nearest colour class
- Colours can then be counted

ALGORITHMS - SEARCH

Spiral | Grid | Parallel | Wall Following | Zone | Random Sampling

Spiral

- Moves around solution space in spiral
- Large area covered
- Quick
- Works best if in centre

Zone

- Splits solution space into smaller chunks
- Drone could visit each zone and do a spin
- Quick

Wall Following

- Follows walls
- Misses centre of solution space
- Object located on wall

HARDWARE

AR Drone

Floating Platform

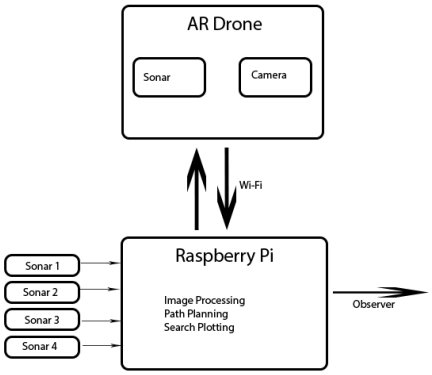
Raspberry Pi

Brain

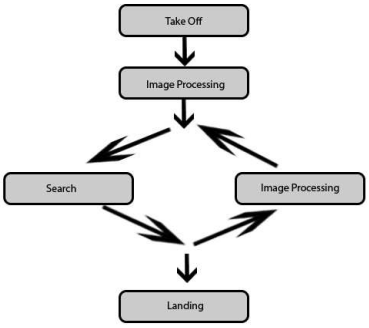
Sonar Sensor

Sensor

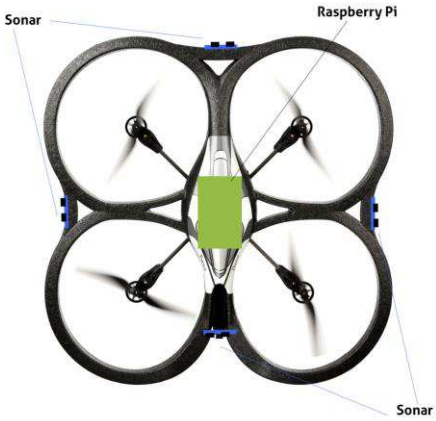
DRONE FRAMEWORK



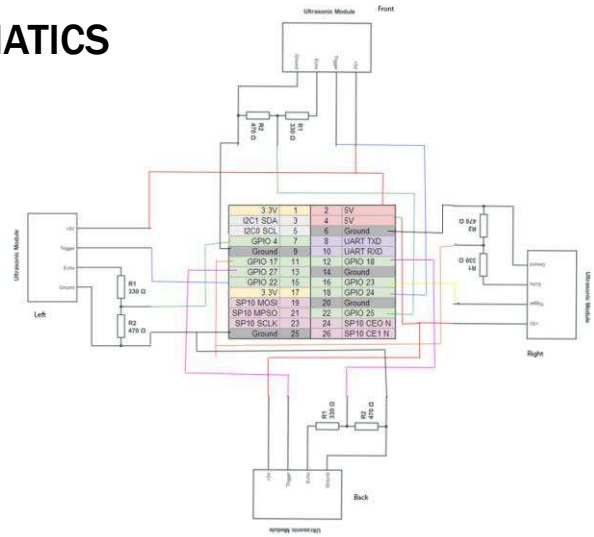
DRONE FRAMEWORK



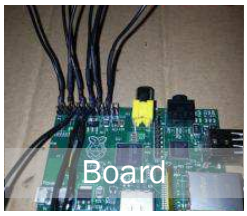
SCHEMATICS

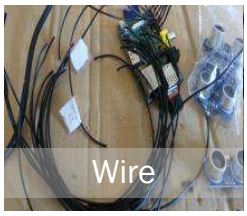


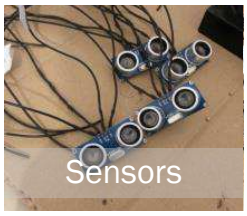
SCHEMATICS

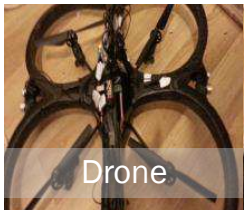


THE BUILD





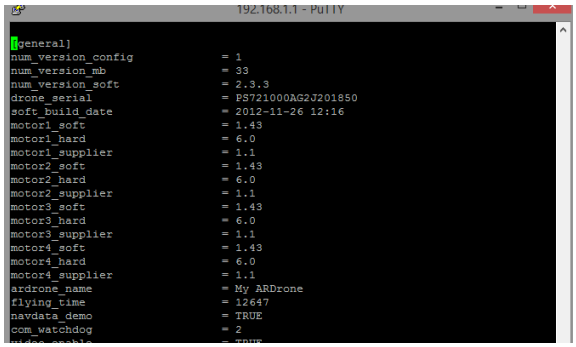




INITIAL HARDWARE TESTS

Weight Issues With Take-off

- Hacking Drone Voltage Configuration

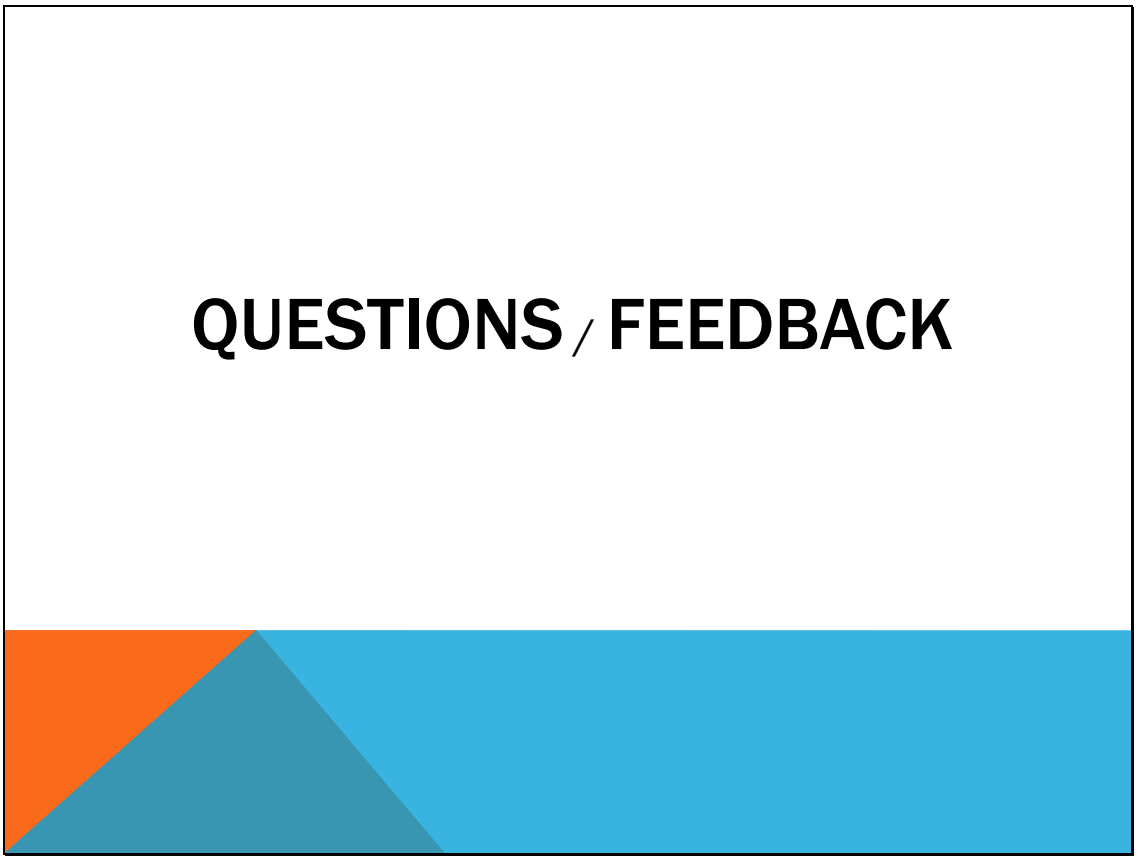


PROGRESS

80%-85%

Next Week:

- Practical Testing
- Results Analysis
- Conclusions, Evaluations...
- Hand In



8.3.2 Feedback

8.3.2.1 Presentation

- Looked Nice
- Don't assume the audience has any knowledge of the subject area
- When questioned knows answers and subject well
- Split up text (too much on some slides)
- Avoid the use of acronyms
- Add more graphics and examples

8.3.2.2 Project

- Elaborate on why you chose the location of the sonar sensors, different configurations, amount of sensors, different needs of algorithms
- How is a good/bad lighting level defined
- Talk about Drone gyroscope and compass
- Why sonar sensors, what other sensors are there
- Add more detail to drone framework
- Test layouts, graphic of layouts
- Visual algorithms, more detailed pros and cons

8.4 Video Links

8.4.1 Concept Video

http://youtu.be/nTY64N47KEo

8.5 Word Count

Counts include headings and appendices. But exclude textboxes, and foot-notes.

Words: 17430

Pages: 119

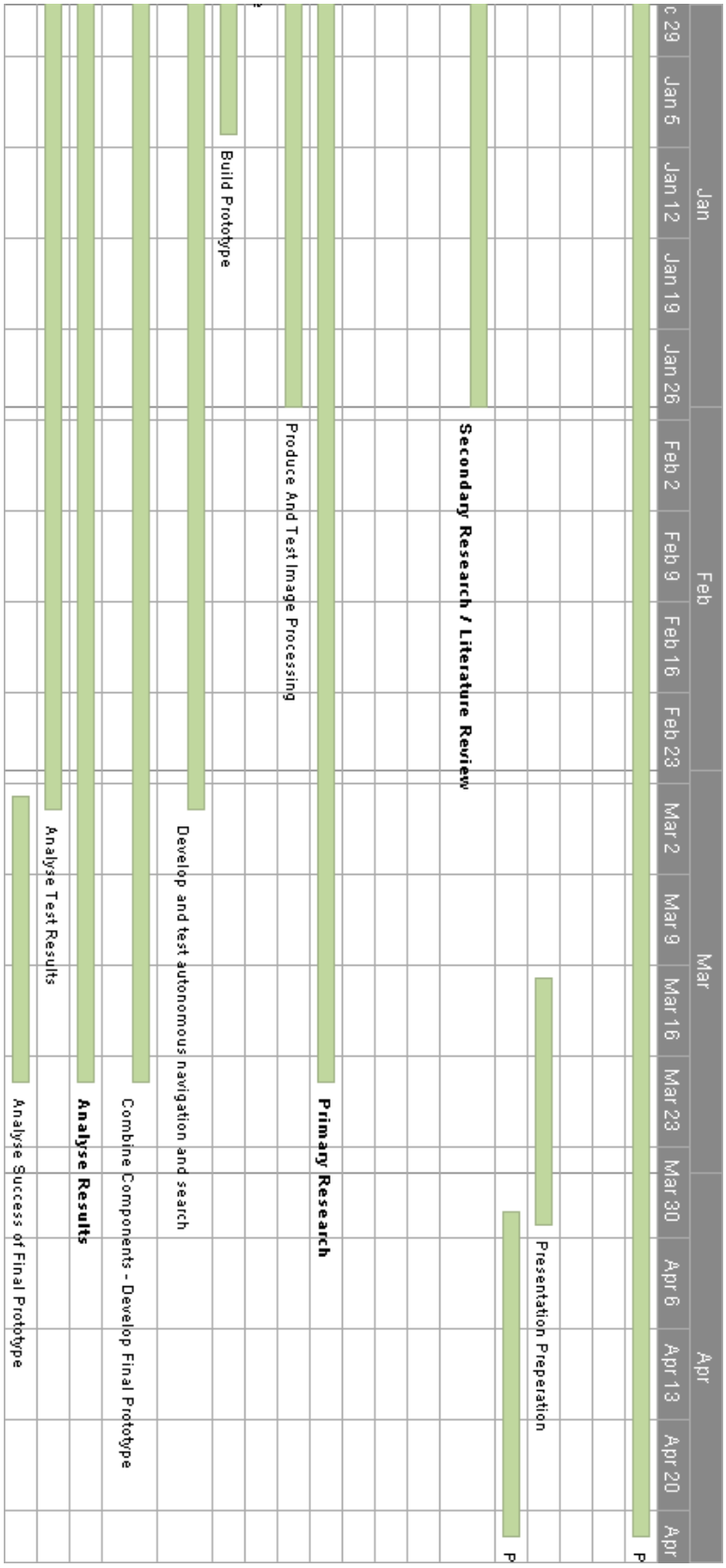
Characters: 93556

Approxamate count excluding bibliography, appendix, contents, figures, abstract, etc.

Words: 13394

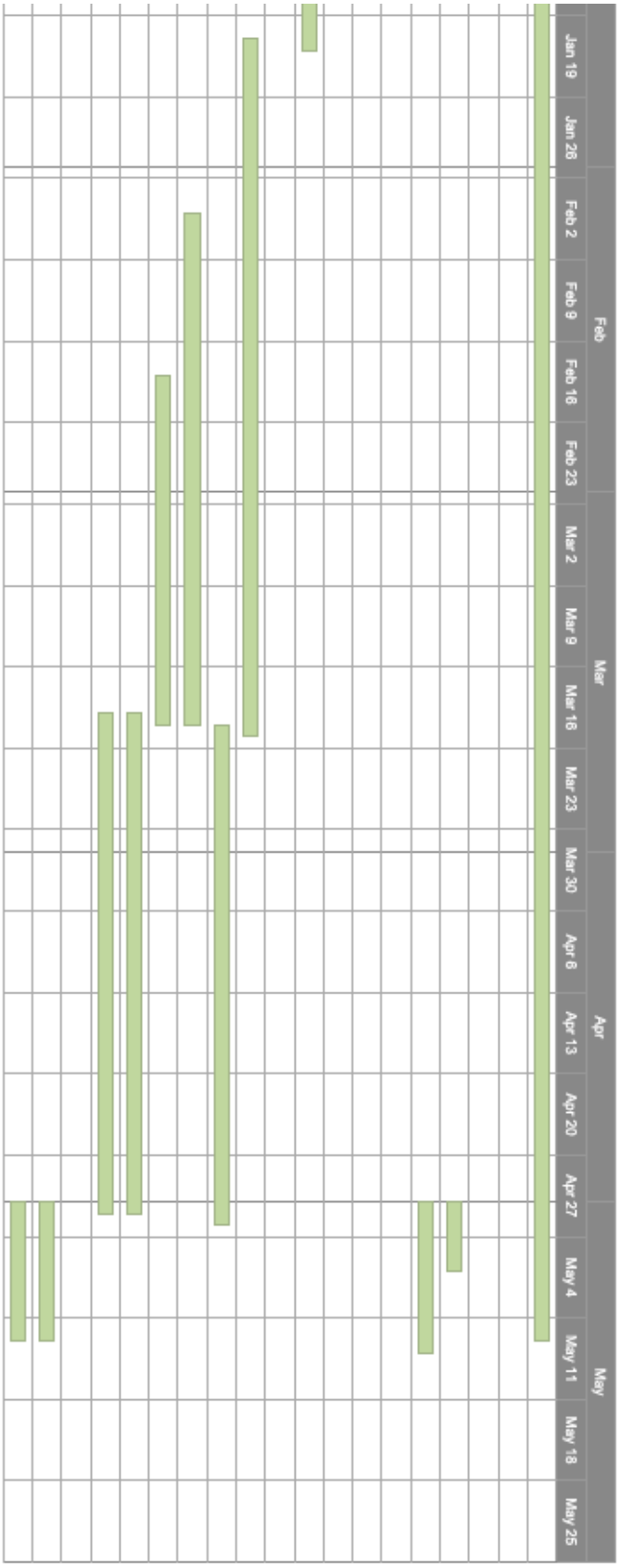
8.6 Project Schedule

8.6.1 Expected

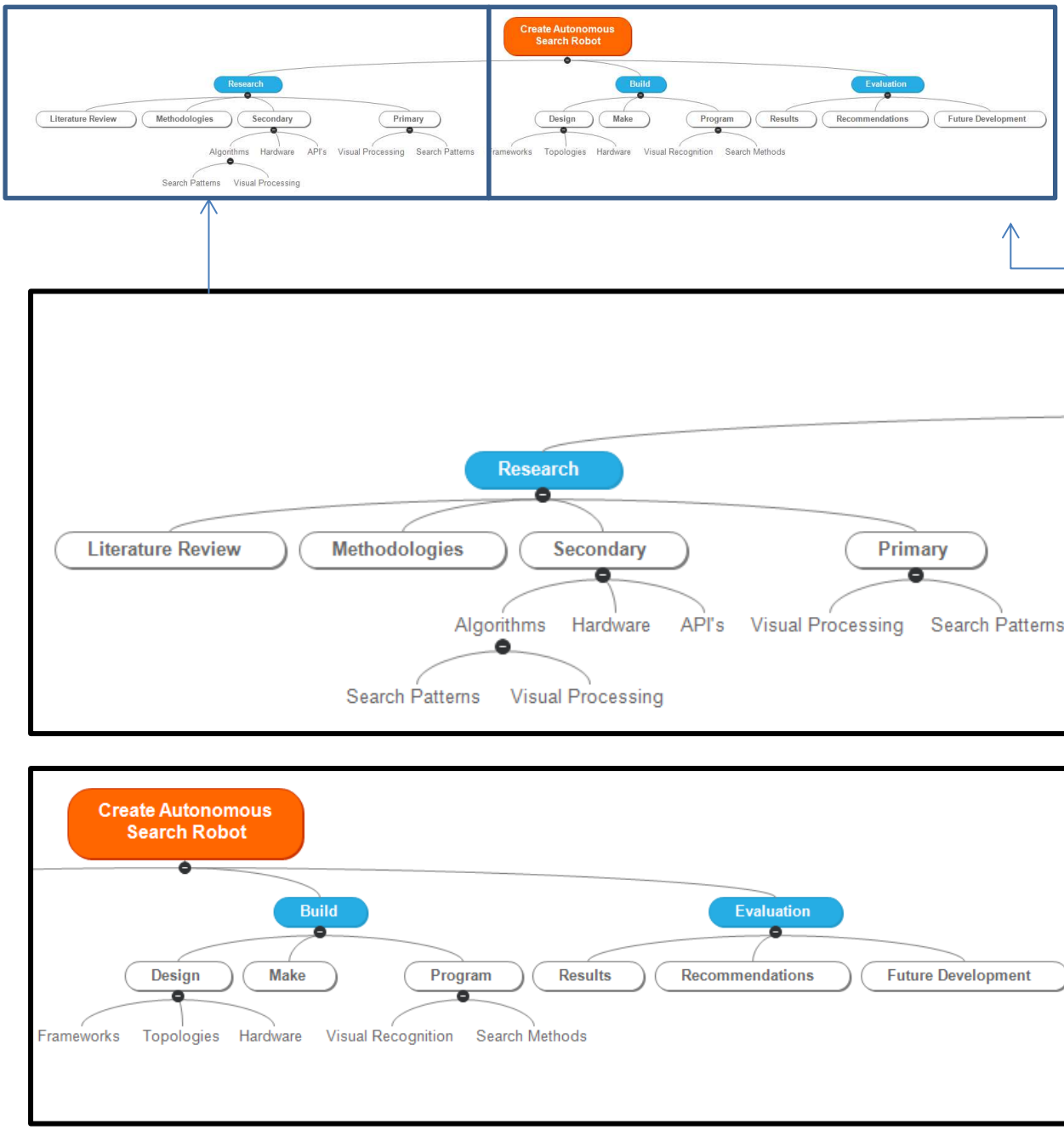

Jan
Feb
Mar
Apr
c 29
Jan 5
Jan 12
Jan 19
Jan 26
Feb 2
Feb 9
Feb 16
Feb 23
Mar 2
Mar 9
Mar 16
Mar 23
Mar 30
Apr 6
Apr 13
Apr 20
Apr
Secondary Research / Literature Review
Presentation Preperation
Build Prototype
Produce And Test Image Processing
Primary Research
Develop and test autonomous navigation and search
Combine Components - Develop Final Prototype
Analyse Results
Analyse Test Results
Analyse Success of Final Prototype

8.6.2 Actual


Task Name
Project Deliverables
A4 Proposal
Detailed Proposal
Presentation Preperation
Final Report Preperation
Lit Review
Ethics and law
Case Study
Research Methodology
Secondary Research
Algorithm research
Primary Research
Design Prototype
Build Prototype
Test Visual Processing
Test Search Patterns
Analise Results
Test Results
Project
Sep 29
Oct 6
Oct 13
Oct 20
Oct 27
Nov 3
Nov 10
Nov 17
Nov 24
Dec 1
Dec 8
Dec 15
Dec 22
Dec 29
Jan 5
Jan 12
Jan 19
Jan 26
Oct
Nov
Dec
Jan


Jan 19
Jan 26
Feb 2
Feb 9
Feb 16
Feb 23
Mar 2
Mar 9
Mar 16
Mar 23
Mar 30
Apr 6
Apr 13
Apr 20
Apr 27
May 4
May 11
May 18
May 25
Feb
Mar
Apr
May

8.7 Work Breakdown Structure



8.8 Risk List

Project Risk List

Risk ID	Headline	Description	Type	Impact	Probability	Magnitude	Mitigation Strategy
1	Project Overrunning	There is a probability that the project could overrun the timescales provided	Direct	3	6%	0.2	Follow work breakdown structure and project schedule.
2	Drone Integration	There is a risk that there may be issues integrating and controlling the drone from the code and using the API	Direct	5	40%	2.0	Use API's to reduce integration issues
3	Sensor Incompatibility	The sensors may be incompatible with the project	Direct	4	5%	0.2	Test sensors
4	Project Integration	The two components of the project (image processing and search patterns) may not be easily integratable	Direct	4	40%	1.6	If not integratable evaluate seperatly
5	Overweight Drone	Components may cause the drone to be too heavy to take off and fly	Direct	5	40%	2.0	Up the motor voltage on the drone, cut down sensors
6	Bad Airflow	Large amounts of airflow through a room may impact the drones flight	Indirect	5	2%	0.1	Ensure windows and doors are closed
7	Collision	Collision with a bystander	Direct	5	5%	0.3	Avoid use in a room with other individuals.

8.9 Project Proposal

Coventry University

Autonomous Navigation and Search in an Indoor Environment using an AR Drone

An Undergraduate Project Proposal

By Erik Barrow

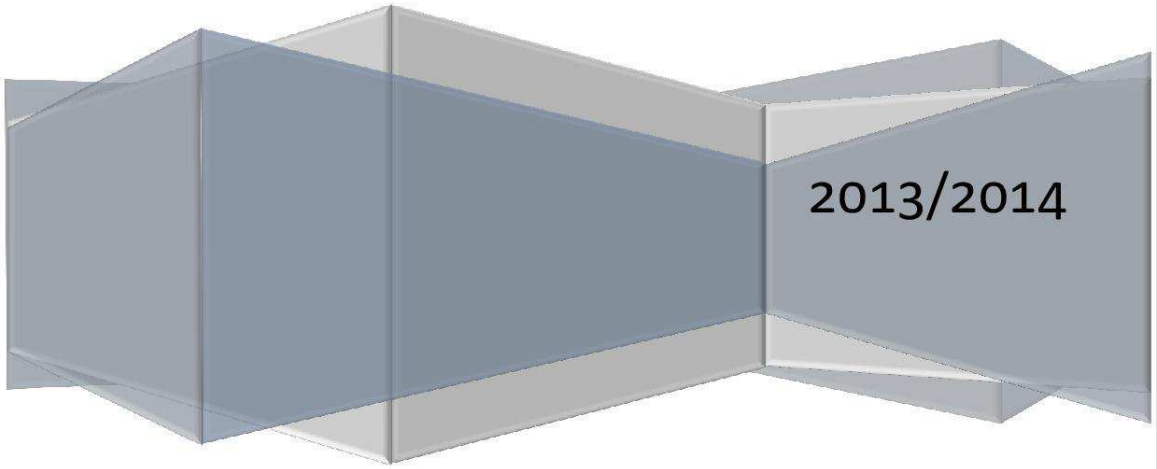


Table of Contents

Student Details 1
Project Details 2
Project Title 2
General Subject Area 2
Project Hypothesis / Research Question 2
Abstract / Summary 2
Aims and Objectives 3
Project Outcomes and Deliverables 3
Motivations 4
Research Methods 4
Data Sources 5
Project Schedule 6
Bibliography 8

Student Details

Surname: Barrow

Forename: Erik

Student ID Number: 3843923

Course: Computer Science

Contact Email: ab3065@coventry.ac.uk / erik@erikbarrow.com

Contact Number: +44(0) 7806773908

First Supervisor: Carey Pridgeon

Second Marker: Amanda Brooks

Project Details

Project Title

Autonomous Navigation and Search in an Indoor Environment using an AR Drone

General Subject Area

AI; Search Algorithms; Autonomous flight; Autonomous navigation; Autonomous search; AR Drone; Autonomous Robotics; Artificial Intelligence; Agents; Robotics; Autonomous

Project Hypothesis / Research Question

Can an AR Drone be programmed to undertake autonomous tasks in an indoor environment? Such as autonomous navigation of a room, while searching autonomously for a person or object.

Abstract / Summary

The field of autonomous robots is still in its early years, with much advanced autonomous robotics being represented in science fiction, many of these science fictions are slowly becoming a reality. From current autonomous robots such as vacuum cleaners all the way up to UAV drones the field of research is vast with possibility. Currently there are minimal UAV drones that operate without the interaction of a human counterpart (such as those used by the Americans, have a pilot somewhere), autonomous drones that can operate without human input can be used in environments that are potentially dangerous to humans or in an everyday environment without the need to a pilot. Tasks for such an autonomous drone could be search and rescue, autonomous mapping, wildlife observation, aerial filming, testing of contaminated areas before human entry, and replacing humans in potentially dangerous situations. While there are few solutions out there, there are not many that both auto-navigate while also preforming a search. Searching also comes in many degrees of difficulty from finding block colours, to an object and face recognition.

This project proposes using an AR Drone (parrot) to produce a solution to an autonomous search robot. Equipped with extra sensors and using AI or agent based programming the drone will be required to roam an indoor environment searching for colours, objects, and people. The project will look into the scope of what objects can easily be detected using the on board camera, as well as other on board sensors, and which image processing algorithms are best for doing this both accurately and quickly on a moving platform. The project also covers looking into navigating an unknown indoor environment using search algorithms, testing and which of a variety of algorithms work best for a hovering UAV. Eventually the drone should be able to given an image of an object or person to find and perform autonomous actions to locate that object or person, without having an altercation with the environment around it.

A prime example of this in science fiction is that of the drones in the Terminator series. While displayed in highly unlikely circumstances the variety of drones, including those that fly, are required to be able to autonomously navigate an environment and locate persons or objects. I plan to replicate this on a lower level, in a more ethical purpose and environment to that of science fiction films. This field of research is also the fuel for current debate on the ethics of how far an autonomous robot should be allowed to operate without human interaction or control.

Aims and Objectives

The overall aim of the project is to research and investigate into methods of autonomous search and navigation, determine the best solutions and to implement these solutions with the use of an AR Drone to a minimum point that the Drone can navigate itself and find a sheet of block colour.

1. A look into the ethics of autonomous robotics, and current legislation and regulations that govern the use of autonomous robotics.
2. Research into current autonomous search, navigation and object detection methods and algorithms.
3. Identify Current case studies and research into autonomous flight
4. Identify the best methods for autonomous tasks based on earlier research for a hovering quad copter.
5. Design a prototype for the AR Drone and acquire the sensors and parts to build the prototype.
6. Develop and code the prototype based of the algorithms researched
7. Conduct field tests searching for an object without colliding with the environment
8. Draw conclusions of the success/failure of the produced solution and make recommendations for future progression of the field.
9. Critically evaluate the project

Project Outcomes and Deliverables

1. An investigation into the ethics and legal aspects of autonomous robotics
2. Research report into methods of autonomous search and navigation
3. Literature review
4. Design and build additional hardware components for the AR Drone
5. Design and build the software system that will run on the AR Drone
6. Working prototype
7. Analysis of research results and recommendations for future advancement.

Motivations

My motivations on this project stem from an interest into artificial intelligence and the use of autonomous robotics. Futuristic science fiction films constantly use depictions of intelligent robots to perform autonomous tasks without input from the user, and while many of these films depict robots becoming self-aware and turning on their creators, I have an interest in what practical applications the technology could have on actual everyday life.

The principles behind the project could also have many to current society. From autonomous search and rescue robots, to autonomous robots working in situations otherwise extremely dangerous to human beings. While the applications could also be turned to warfare, which is a highly debated ethical issue, there are many positive uses for the technology.

Research Methods

To investigate into the area of autonomous flight I will use a mixture of research methods. I will start with secondary research into current methods of autonomous search, navigation and image processing. There will be a multitude of different algorithms that could be used in my project, and I will need to determine which of these are best for my project.

I will then look at any current studies on autonomous robotics and autonomous flight. I will look at how those projects where conducted and what methods they used. I can use these examples to help me predict any pitfalls in certain methods or algorithms that may affect my project. This will all be done in a literature review early on in my project.

My primary research will use the methods of autonomous navigation and search that I found in my secondary research. As my research is mainly an experimental study I will use an AR Drone to develop and test some different methods of automation for the drone and adapt these methods to provide the best result for the hovering platform (AR Drone) . I will do this by constructing a prototype drone with added sensors that I deem will be necessary for the project to operate, such as sonar and thermal imaging cameras. I will conduct some observational studies of the drone while it is stationary to test the image processing and autonomous search aspect of the project to see what I can easily detect with the built in camera, and then I will conduct the same observations to see how well these methods work on the drone while it is hovering and or moving.

I will also experiment with the different autonomous navigation methods that I find in my literature review, to determine which the best for a hovering platform is. I will do this through experimentation, physically testing and observing the results of the navigation combined with search algorithms to see how well the drone can navigate an entire room autonomously. With the navigation, search, and image processing decided and tested I will then combine and test them for a final prototype.

My research will be mainly quantitative research as there should be measurable data produced from my experiments. Although there could also be some qualitative results as the nature of the AR Drone could give some vague results when interacting with the environment, possibly producing different results for the same task.

Data Sources

There is a large pool of sources I can use to gather reliable data for my project. These sources of data come from many places, from subject databases, product documentation and API's, as well as the data I produce as a result of experiments.

In my research I will use several subject databases to find papers and journals on autonomous robotics, examples of the databases that I plan to use are the ACM Digital Library [10], and the IEEE Xplore database [11]. Both of which I have found to have reliable and a large quantity of research papers that I can use to my advantage. I could also use the Science direct database [12]. All three databases have provided me with some example papers that I will use during my research, such as [1] "Autonomous Navigation, object detection and retrieval in an Indoor Environment" and [2] "CYCLOPS: A mobile robotic platform for testing and validating image processing and autonomous navigation algorithms in support of artificial vision prostheses"

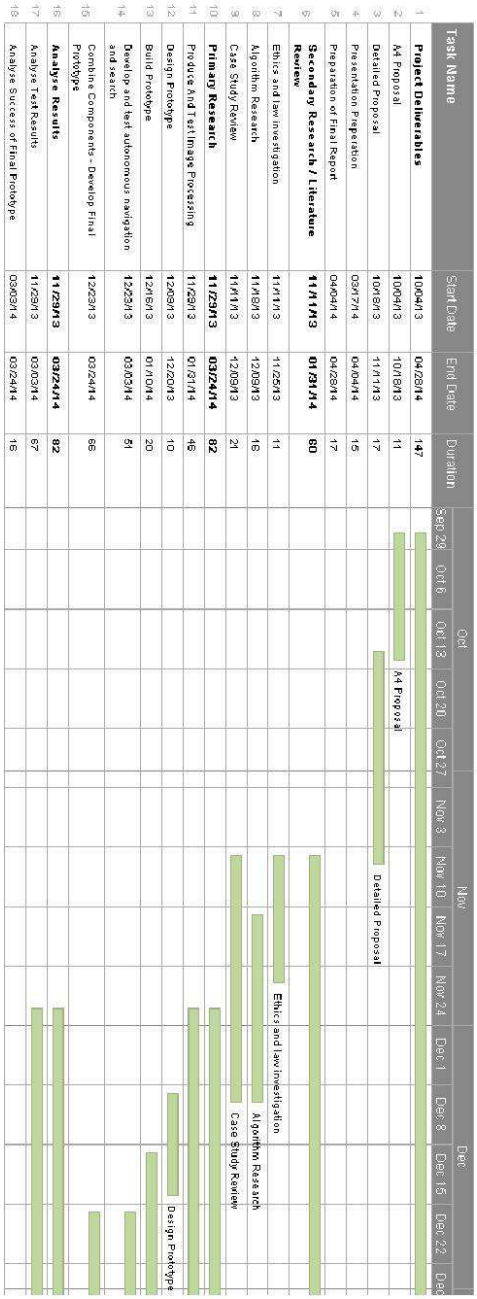
The Lancaster library [3] also has some books on autonomous robotics that may be useful, looking at both the practical aspects of the research field but also the ethical implications of autonomous robotics. Books and journals such as [4] "Robotics and autonomous systems" and [5] "Robot ethics the ethical and social implications of robotics".

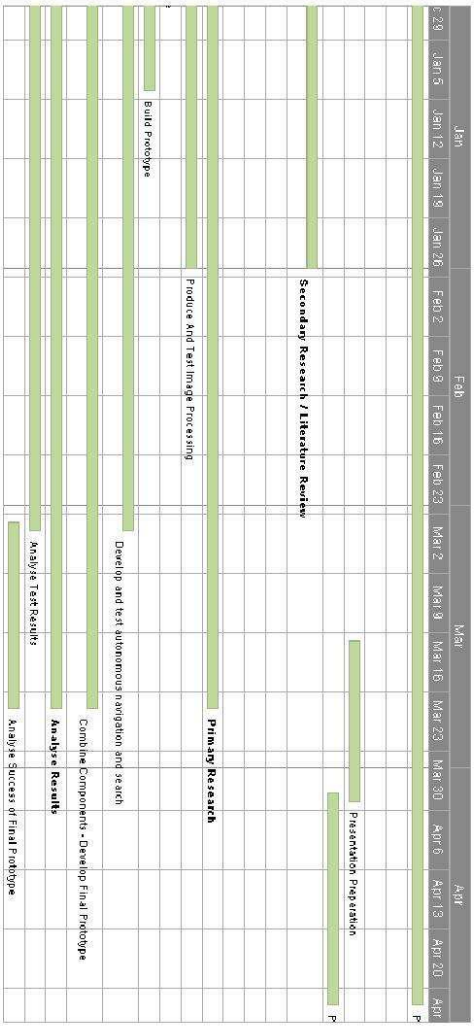
In production of my prototype I will need to use a variety of product documentation for the products I am using as well as the API's that I will be using to aid my programming. There will also be a variety of developer forms I will use to help me understand the API's and for extra documentation. The AR Drone has an API website [6], which provides the API code, as well as all the documentation that the company produces and a developer form for developers to share their own documentation and experiences with the drone.

I am also planning of using a raspberry pi to attach sensors and run the automated code and to interface with the drone. For this I will need the documentation for the raspberry pi [7].

More papers that I will be useful could be [8] "Opportunities and challenges with autonomous micro aerial vehicles" or [9] "Utilization of machine learning methods for assembling, training and understanding autonomous robots".

Project Schedule




Jan
Feb
Mar
Apr
Jan 5
Jan 12
Jan 19
Jan 26
Feb 2
Feb 9
Feb 16
Feb 23
Mar 2
Mar 9
Mar 16
Mar 23
Mar 30
Apr 6
Apr 13
Apr 20
Build Prototype
Produce And Test Image Processing
Secondary Research / Literature Review
Develop and test autonomous navigation and search
Primary Research
Presentation Preperation
Analyse Test Results
Analyse Results
Combine Components - Develop Final Prototype
Analyse Success of Final Prototype

Bibliography

[1] Achal D Arvind, Shashanka L (n.d.) Autonomous Navigation, object detection and retrieval in an Indoor Environment

[2] Wolfgang Fink, Mark A. Tarbell (2009) CYCLOPS: A mobile robotic platform for testing and validating image processing and autonomous navigation algorithms in support of artificial vision prostheses

[3] Lancaster Library (n.d.) http://locate.coventry.ac.uk

[4] Amsterdam North-Holland (n.d.) Robotics and autonomous systems

[5] P. Lin, K. Abney, G. Bekey (2012) Robot ethics the ethical and social implications of robotics

[6] AR Done API (n.d.) https://projects.ardrone.org/projects/show/ardrone-api

[7] Raspberry Pi Documentation (n.d.) http://www.raspberrypi.org/technical-help-and-resource-documents

[8] V. Kumar, N. Michael (2012) Opportunities and challenges with autonomous micro aerial vehicles

[9] Hartono, P. (2011)Utilization of machine learning methods for assembling, training and understanding autonomous robots

[10] ACM (n.d.) http://dl.acm.org/

[11] IEEE Xplore (n.d.) http://ieeexplore.ieee.org/Xplore/home.jsp

[12] Science Direct (n.d.) http://www.sciencedirect.com/