A Robust Network for Cooperative Drone Swarm Mapping

Hugo Robinson

May 2022

Abstract

With the use of drone swarms on the rise for cooperation and conflict, they will inevitably be deployed into hostile areas where drones that enter, won't return. Thereby reducing the effectiveness of the swarm as a whole and negatively impacting the swarm's ability to complete assigned objectives. The purpose of this dissertation is to suggest a solution to the problem outlined above. Following a literature review of current path-finding, localization, mapping and networking techniques, the A* algorithm was originally chosen for the drones to use, however, this has proven ineffective when pathing through a large, dense graph, therefore a unique path-finding technique is proposed called radial deflection that takes inspiration from A* and RRT. The drones map the environment through a simulated LIDAR sensor, LIDAR is chosen for its practicality in simulation over computer vision, the data provided by the sensor is then used to create point cloud maps of the environment which are visualised using the pygame library. Further to this, a system of information dissemination is proposed to create a robust network for the swarm where neighbouring drones share their point cloud maps to increase the overall knowledge of the network and reduce the uniqueness of a single drone. This system is proven to be effective with up to a 68% increase in map exploration whilst under attrition than a network that does not have information dissemination. This leads to proposed future improvements of the network and system as a whole.

I certify that all material in this dissertation which is not my own work has been identified.

Contents

1 Introduction

This project uses simulation as a basis for developing an algorithm to enable swarms of drones to act cooperatively in mapping a 2D environment. With the drastic increase in drones and drone swarms being used for surveillance and imaging [1], [2] in both the civilian and defence sectors, [3] the need for more robust and reliable networking between these drones becomes stronger. However, until recently these drones have been expensive and, generally, not connected to a network.

Within the field of robots, path-finding through environments produced major results in the 60s [4], showing that a robot can path through and reason about its environment. Drones as we know them today did not begin to take shape in the civilian sector until the 2000s with the previous 40 years dominated by large military surveillance drones [5]. With the first commercial drone permit being issued in 2006 [6], unmanned drone technology had now moved out of the realm of defence into the civilian sector. However, these drones may be unmanned but were not automated, they needed a human to operate them and were still not part of a network. Throughout the 21st century, drone swarms have developed and been used in disaster relief [7], [8] and area mapping with swarm technologies starting to be used in the defence sector [9], [10]. This has brought drone swarms to the front of both news and research.

The underlying principles of this project are the 2D environment itself, how the drones are simulated, the drones' path-finding, how the drones sense the environment and map it and how they do this cooperatively. The environment and how the drones are simulated will provide the basis of the project dictating both how the environment is built and how the drones can move through it. With extensive research on different automated path-finding techniques [11], there are many avenues to explore, no pun intended. With path-finding problems being researched as far back as 1926 [12] by Otakar Borůvka, to the famous Dijkstra's Algorithm [13], to the A* Algorithm and its descendants. Sensing the environment can be done in several ways, from computer vision, combined with machine learning [14] to the more mechanical approach of LIDAR [15]. With further developments, drones can be equipped with small, lighter and more powerful sensing equipment, improving performance. Mapping an environment and the way it is represented is an equally important area for drones. With the LIDAR system a point cloud map can be generated, storing data as points within an array as opposed to storing stitched together images from a computer vision-based approach, making it a simpler representation, however with computer vision the drone may be able to do more complex tasks such as target identification using machine learning, a skill that LIDAR cannot recreate. For a swarm to work together, some form of networking must take place, standard networks have set nodes and connections, allowing data transfer optimisation. Whereas drone networks have moving nodes which is a distinct difference, an issue tackled through the development of Ad-Hoc networks [16].

In this report, a summary of the literature surrounding these points is examined and synthesised along with an outline of experimentation and success for the further project. An overview of the project as a whole is then described, including problem complexity and system design. This is further broken down into detail showing a more in-depth look into each area of the system with challenges encountered and solutions found, contributing to the wider picture of the project. The results of the experimentation are then detailed before further ideas for future projects and a conclusion.

2 Literature Review and Project Specification

2.1 Path Finding

Path-finding is a vital area of research for any automated vehicle[17] [18], with many different techniques it is important to determine effective methods based on what the drones are going to do.

2.1.1 Search-Based

Search-Based algorithms take the approach of checking the points immediately around an entity to determine the next best move it can make [13]. These algorithms are common and effective in their output, with some guaranteeing optimal paths [19]. The two most well-known examples of search-based algorithms are Dijkstra's algorithm and A*, with the latter being built on top of the former.

In the paper [20], the authors model a drone as a point on a 2D grid. This grid contains walls to make a simple maze for the drone to path find around. The author's results show that Dijkstra's algorithm finds the shortest path in this scenario and therefore is a useful algorithm for simulating drone movement on a grid. However, this works best with known environments and not environments that the drone is currently exploring. This reduces its effectiveness as the drone will have to find a new path when it encounters a new obstacle.

An improvement on Dijkstra's algorithm is the A* algorithm, where a heuristic is used in the search and therefore does not have to search as many cells as Dijkstra's to find the optimal path [21]. This is a significant improvement over Dijkstra's, especially in dense graphs such as a grid. However, it will still have to scan many cells in the graph and is not effective in an unknown environment. This can be seen when A* encounters an A* trap and must scan many cells to be able to escape this trap. An example can be seen in Appendix 1.

2.1.2 Sample-Based

Further to the search based approaches shown above, there are sampling-based approaches. These set waypoints or nodes in the environment and produce an unobstructed path from the original point. Therefore the algorithm just has to path between these nodes, which are all unobstructed from each other, decreasing the time it takes to find a path [22].

Rapidly-Exploring Random Trees are the most well-known version of sample-based path-finding, where the algorithm randomly selects a visible node up to a set distance away. The algorithm quickly explores spaces as it is more likely that the node will be selected in an unexplored space than an explored one [23]. An example of this can be seen in appendix 2.

An improved version of RRT is RRT* [22] where each selected node searches in a radius around itself to connect to nodes near it. This allows the tree to be restructured to produce a shorter path, this, in turn, creates a more optimal path for the drone.

However, all of these algorithms share the same issue, of working best in known environments and therefore being more suited to the later stages of drone exploration where more of the map is discovered and the information is disseminated between the drones.

2.2 Simultaneous Localisation and Mapping

Simultaneous Localisation and Mapping (SLAM) allows a drone to both map its environment and know where it is within it [24]. For a drone to achieve this, it must be fitted with hardware that can 'see' the environment around it, one such piece of hardware could be a Light Detection and Ranging (LIDAR) system that is used to detect surfaces. These measurements can then be used by the drone to create a point cloud map of its environment where each detection is saved as a point in space [25]. For Binary, 2D maps, which is what this project will focus on the point is saved onto a Binary Occupancy Map for ease of representation [26].

The combination of SLAM and path-finding will allow a drone to explore the environment it is placed in. As new walls are discovered it will have to re-plan its path around these findings, this can be done every set number of time steps, every time it encounters a wall or a combination of the two.

2.3 Networking

For a drone swarm to work together, an effective network must be implemented, however, these networks are different from traditional networks as the nodes move around, lose connection and create different routes for data to transfer. This places new requirements on the network, for it to be able to self organise and have no predetermined structure. Mobile Ad-Hoc networks are the answer to this issue [27]. This network structure is designed to allow nodes to disconnect and reconnect with each other as they move through an environment, perfect for drones.

Ad-Hoc networks can be categorized into three styles: leader led, decentralised and hierarchical. Each with different advantages and drawbacks.

A leader-led network has one drone controlled by a human operator that directs the swarm towards an objective, each of the other drones simply follows along. The main benefit to this is that it is a simple way to increase drone coverage from a single human, however, it provides no automation or complex task handling.

Decentralised networks are networks where each node/drone shares equal responsibility in the network, communicating only with its immediate neighbours [28]. This reduces the time and effort of a human operator as they do not need to specifically direct a swarm as the swarm will be given general task directions and will itself distribute responsibility throughout the swarm. Equally this style of network reduces complexity as the swarm works with simple communications between neighbours leading a complex system to emerge from simple interactions. However, this does restrict the type of tasks the swarm can carry out as if the objective cannot be encoded into simple interactions between neighbours then it loses its advantage.

Finally, hierarchical networks split the swarm into smaller units that work as a smaller swarm, with one drone acting as a swarm leader, disseminating objectives to the drones beneath it and communicating with the other leaders to form one larger network. One significant benefit of this type of network is that the leader drones can reduce unnecessary communication by filtering out objectives that the drones beneath it do not need to worry about, therefore these drones would act in a similar way to routers [29]. However, there is a large drawback to this type of network, where if the leader drones are compromised in any way whole areas of the network cannot receive orders or share their information with surrounding drones. This is particularly true in dangerous environments where there is a high likelihood of the drones sustaining damage.

Overall multiple styles can be used, with various advantages and disadvantages and a combination would likely provide the best results.

2.4 Success Criteria

With the combined data from the drones taking the form of a binary occupancy map, I can compare this 'master' map to the original and determine how much has been explored, this percentage will provide an effective comparable basis for experiments.

3 Design

The project is split several distinct parts that allow the whole thing to function correctly. These areas are the simulation itself, drone SLAM and drone networking. With each of these areas building on top of the previous.

3.1 Complexity

There are certain areas of this project that provide significant complexity to the problem. The environment must be able to support drones moving through itself as well as walled-off sections and be easily consolidated into data that can be utilised by the drones. In close combination with the environment,

the path-finding ability of the drones is a complex problem, with multiple options available to be used with varying advantages and disadvantages. Each drone will be able to communicate with each other and with the ground station. These communications may also not be map wide and therefore each drone must be able to work out which drones it can communicate with at any given time. Alongside this, a standardised communication protocol must be established to efficiently share data between drones. For the drones to become a swarm they must network with each other, this is especially challenging as it drastically increases the overhead and processing for each drone which will need the ability to marshall and unmarshall communications with each other. Equally the design of the network itself causes a drastic increase in complexity from many individual drones to getting them to work together.

3.2 Simulation

The backbone of this project is the simulation of the environment and the drones. With all components being run within this simulation it must represent the way a drone in the real world may process its environment. A simulation also provides distinct advantages for test purposes. As it allows for immutable parameters, as the project does not revolve around how drones react to different weather patterns, or how their sensors react to different visibility these variables are not included within the simulation providing more reliable data on the drones and networks themselves. These reasons also extend to the internals of the drones, with different drones having different networking protocols and many not being able to network with each other. Whereas with a simulation the networking can be custom built to be lightweight and exactly what is required. Finally running a simulation is much less expensive than buying a large number of drones and learning how to operate them, further solidifying the decision to simulate the drones, as opposed to using real ones.

A couple of possibilities were considered for how to simulate the drones and environment. The requirements for this technology are cost, practicality and customisation. Firstly C++ was considered, with its widespread use in developing games and simulations alike it is a good choice. Alongside this, the speed C++ can provide compared to other languages reduces the simulation time, particularly for lots of entities or large amounts of processing.

Taking a different approach to a set language, Unity was considered. Unity is a game engine that allows users to create games, from building and texturing worlds to developing dynamic events. Unity provides fantastic visualisation as it is easy to develop and view the environment, it also would allow for 3D environments much more easily than C++. As Unity is specifically designed for game development, which in a sense is a simulation, it provides pre-built infrastructure for developing the basics of the simulation, alongside the ability to integrate personal code into the entities such as movement and communication.

Python was also considered, with its vast array of libraries and ease of programming, it provides a flexible language to work with. The Pygame library allows the user to develop games and simulations only using python and provides many functions to help with this.

After looking at all of these possibilities, Python was chosen as with my personal experience with it, it will reduce the time and complexity of programming the project. Also, the extensive libraries and support for the language mean that it is well suited to the requirements of the project.

3.3 Assumptions

As this is a simulation, various assumptions must be made about how the drones and environment operate and interact with each other.

Current technologies for LIDAR may be very good but are not always perfect. This can have the effect of leaving artefacts within the map or not placing some points on the map at all. Artefacts left on the map have the effect of creating blockers that do not exist, slowing down path-finding algorithms or blocking off certain areas from being explored. Further to this if some points are not placed on the map due to errors within the LIDAR system the drones may collide with an obstacle or, specifically for the simulation, they may get trapped within a wall because they did not detect the point until the wall was

behind them. In order to simplify the problem, in the first instance at least, I have chosen to make the LIDAR system perfect, with no errors or noise.

Much like most real systems, the data collection or transmission may not be perfect. This can be easily seen with localisation for autonomous movement. This can be achieved through self-localisation by viewing its surrounding environment and attempting to map its movement. This is called odometry, however, if either of these systems fails the drone will completely lose where it is within the world and then will be unable to path find itself effectively, again causing collisions with the environment. A similar occurrence can happen with systems such as GPS, especially within buildings where the connection can be disrupted or lost, therefore blinding a drone to its position until it is reconnected.

Each drone can communicate with other drones and with the ground station to share data. As is relevant with any network and even more obvious with wireless networks that have some dropouts and errors with data transmission. This can cause incorrect maps and data to be transmitted or none at all.

In a similar vein to odometry, drones have to contend with the weather of the environment there are in, this can cause errors in flight and odometry by moving the drone around without it being able to accurately record where it is and therefore has similar issues. Therefore there are no effects or simulations of drone flight within the simulation and drones move from one node to the next without any deviation. This is because these simulations can be expensive to run reducing the effectiveness of the simulation for no gain.

The environment that the drones explore will be static and the environment will not change as the drones move. This is to reduce the need for retracing a path during the drone's flight time, this reduction allows the simulation to run faster. The reduction in overall complexity this provides allows for a reduction in scope.

Overall this project is built on top of several assumptions that have been laid out here, these assumptions reduce the scope of the project allowing for a more targeted approach to the question. Without a reduction in scope, the project could suffer from becoming too large and not developing any one direction in sufficient detail.

3.4 Environment Structure

The environment will be a 2D grid using the pixels of the pygame area as the gird format. Therefore the environment acts as a dense graph for path-finding. The walls of the building will be different colour pixels within the grid that can be seen by the drone and avoided. The environment as a whole will be static whilst the drones are attempting to map it, however, the map as a whole will be hidden from the drones and they will only know what they have mapped previously. Therefore this previous mapping will always be up to date for the drones. As the map is 2D the structure of the buildings is just a top-down view of the walls. Doors and windows are indistinguishable from one another as they both act as openings to the building from a top-down perspective. Furniture may be added however they will also take the form of walls within the building. An example of this structure can be seen in figure 1

3.5 Code structure

The program will follow an object-oriented design where each distinct part will be its object. This allows for many instances of the drone object and one overarching instance of the environment object. The main class will initialise all of the other classes and begin the simulation. The environment class contains the location of everything within the environment, such as the map and the location of the drones. A base station class acts as the start point for the drones and will combine their maps into one master map that is produced and will direct the network. The drone class contains all the methods that each drone will need and allows for drones to be added or taken away during the simulation. The sensor itself will be a separate class, shared by all of the drones as it acts with the environment to produce the data for a local map.

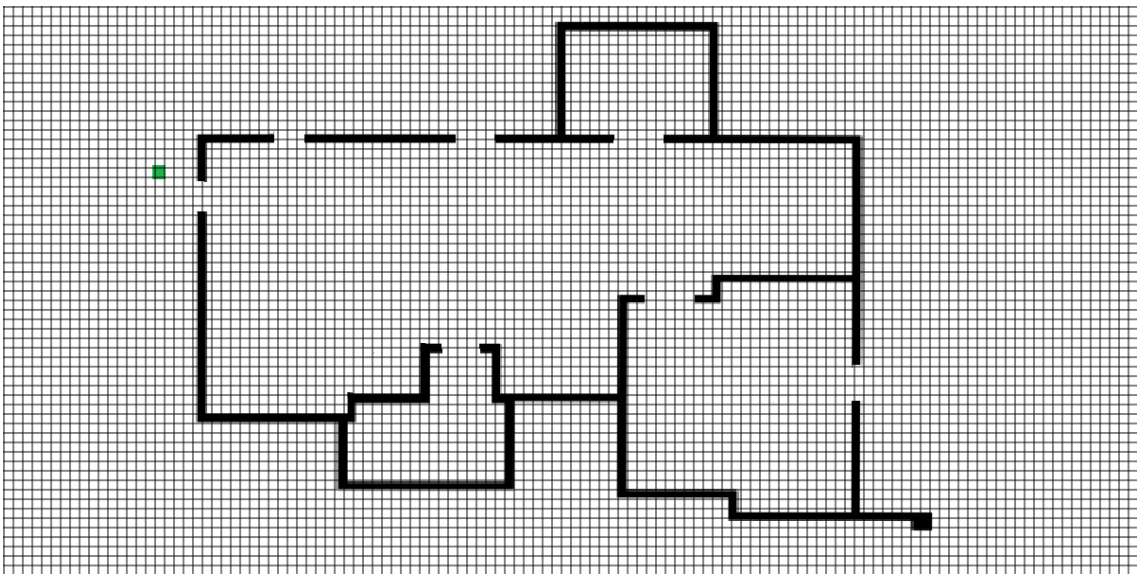

Figure 1: A diagram of the grid format, with a green done and black walls of a building.

3.6 SLAM

For each drone to be able to move they must each have the ability to map the environment and find a way through it. As discussed in the literature review there are a couple of ways that modern drones view their environment however combined with the decision to make the environment 2D and ease of implementation LIDAR is simulated here. The LIDAR provides a 360° view around the drone, in reality, these LIDAR sensors are very large and must be mounted on large drones, however, the decision has been made to simulate a 360° LIDAR on a small drone as it is the best-case scenario, a technology that may be available soon and reduces the time taken to develop an unnecessary part of the program.

Path-finding is a major area of the project as it allows the drones to move around the map and avoid the obstacles they discover. As discussed in the Literateur Review there are many different algorithms for path-finding, each with its advantages and disadvantages. The decision was made to use the A* algorithm as it is easy to implement, works well on most graph structures and is the basis of other path-finding algorithms. Through some original testing, this worked well when the map was empty or full. However, in the interim period where the map was being discovered, the drones had to re path every time they came near a wall. This in turn drastically increased the time it took to reach its goal. Therefore the aim switched to pathing to the goal in short steps that would not take the A* algorithm very long to find. Therefore a combined A* and way pointing system is then implemented. Where the drone sets a waypoint 100 pixels away and between it and the goal position, it will then path to this interim position and set a new one until it is at the goal position. This reduces the time each drone spends finding a path to the goal position.

Without real wireless communication, it becomes difficult for each of the drones to communicate with each other as it must somehow be simulated. To begin with, each drone had a list of each instance of all the other drones, this allowed it to know where each drone was by accessing its parameters, which hold the drone's current position. The drone would then iterate through this list, comparing the distance to each drone and if another drone was within this distance it would be able to communicate. The communication aspect is a self-made protocol that details the drone's local map alongside any relevant information and is done by calling the method from the second drone and passing the information as parameters. Although this would work, it would be a messy solution, with each drone intimately knowing each other the simulation would stray further from the real world. To keep this continuity with the real world and allow the drones to check if they are close enough to communicate, I moved these lists of drones to one list and to a place that would know where they all are in the environment. As each drone is part of

the environment and interacts with it, the environment class 'knows' where they all are. Therefore when a drone wants to communicate with another drone, it queries the environment class for this list, checks the distances and communicates as previously detailed. Although functionally similar this decoupling of each drone is more representative of the reality.

3.7 Networking

The network design will be a hybrid of a distributed design and a hierarchical design, where all of the drones are equal, all mapping the environment in the same way and passing communications in the same way. With each only talking to their immediate neighbours and the ground station. This is where the hierarchical element comes in where all the drones are initialised and directed by the ground station, this is also where all the maps from each drone are combined into a final map. Equally the drones can be directed by the ground station to improve the mapping of the environment. One such direction could be to start by going around the outside of the map, finding the outside walls, and then using this information to go through all entrances at once.

The proposed solution to network robustness is to share information whenever possible. This means as soon as two drones can communicate they share their explored maps. This means if one drone is lost others will have a copy of its explored map reducing the uniqueness of individual drones, and improving the reliability of the swarm.

3.8 Possible Experiments

With multiple network styles that can be used, implementing both a hierarchical network and a distributed network can provide data on which one is more robust and can scan the environment faster.

As information must be passed through the network a change in how the drones communicate with each other will change how the network operates and data can be collected on distances between drones or how much they can communicate.

A swarm uses a lot of cheap drones, therefore numbers count to reduce costs and a critical mass can be determined where an increase in the number of drones provides no additional advantage in information gathering.

4 Development

4.1 Drones

The development of the drones is based on encapsulating all of their functionality within a single class, therefore representing the real-world dynamic of using drones. These have several parameters. Each drone has its identifier, this number is allocated as it is initialised and can be used to identify different drones from each other. One of the most important parameters that each drone contains is its local environment, this is its representation of the environment that has been explored by itself. It is a simple list of tuples that represent the data points scanned by the LIDAR sensor. Therefore this local environment represents the point cloud map of the explored environment. As one of the assumptions is that each drone will have perfect knowledge of where it is, this is represented as a position parameter that is simply a vector tuple of its coordinates and can be easily updated or queried when moving. The drone also has many parameters dedicated to movement and pathing, the drone creates a single goal position and path towards it whenever it needs to. This goal position is randomised and can be anywhere within the boundaries of the environment, however, this undirected approach is not optimal for exploration.

Each drone also contains the instances of both the sensor and ground station to allow it to scan the environment and communicate directly with the ground station.

4.2 Environment

As the environment itself is the base of the whole project it is developed early to support everything else. The environment is based around using pygame displays which use pygame surfaces for both visualisation and some interaction. An image of a floor plan is used as the base of the environment. With black walls and white background, the difference is easy to differentiate. This is then copied to a separate parameter that will be the surface that is shown to the user when the drones are mapping. As the environment class contains everything that exists within the environment, it also contains a list of all of the drones and therefore can track them. Therefore the environment acts as a global class with both the complete map and the knowledge of all the drone locations as these entities become part of the environment.

4.3 Visualisation

Pygame surfaces are used to represent an object on the screen, therefore it encodes the image underlay of the map. Pygame surfaces contain some methods that are very useful to this project, with the first being fill, where the whole map can be filled with a single colour, therefore when shown on screen the map can be all black, hiding the map underlay from the user. The copied environment is used to display the known environment to the user and therefore a method in the environment class is used to do this by setting a coloured point of the drone positions and their local maps, red for a wall, green for a drone. This produces a map that can be seen in figure 2.

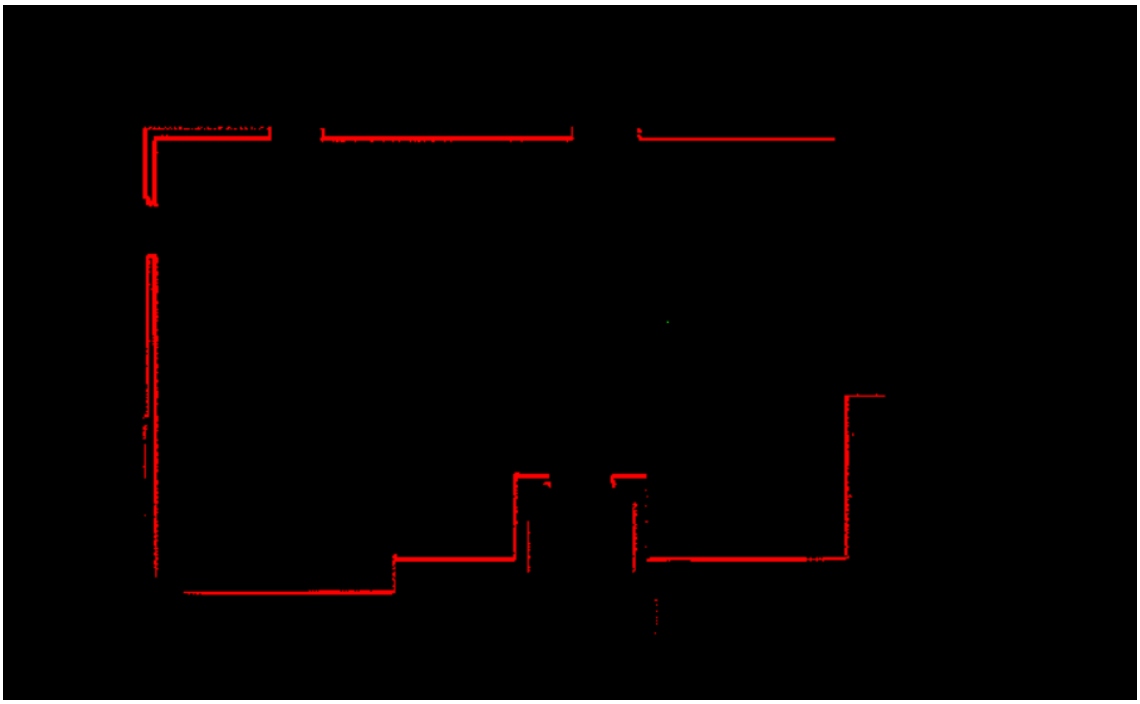

Figure 2: An image of a partially explored map, being explored by a single drone, the drone is the small green dot in the middle of the screen.

4.4 Path Finding

Path-finding for this project provided a unique problem because multiple drones had to path find in an unknown environment simultaneously on a single processor. Therefore both accuracy, adaptability and speed are required to find a solution.

Each drone must reach a set goal position within the map which may be close to it or far away and is set either randomly or directed by the ground station. Once a drone reaches its goal position it is removed

from a list of positions it must reach and a new position is set, therefore a drone can continue to move through the environment mapping larger areas.

4.4.1 A* Path Finding

As discussed in the literature review section, the A* path-finding algorithm was selected as it guarantees the shortest path to the destination and is therefore optimal. However, the capability of A* is reduced when it is not provided with a complete map at the start as it will find an optimal path but will not know of the obstacles in the way and will therefore path through them. This can be seen in figure 3. using purely A* may work later in the simulation when more of the map is discovered and therefore each drone can optimally path around walls. Equally, this takes a long time as the drone must check thousands of pixels between itself and the goal position, it has to do this every time it finds a wall in the way and therefore must re-path many times before it reaches its goal position. This causes the whole simulation to stop, meaning it takes a very long time for the simulation to complete, particularly when multiple drones are doing this at the same time.

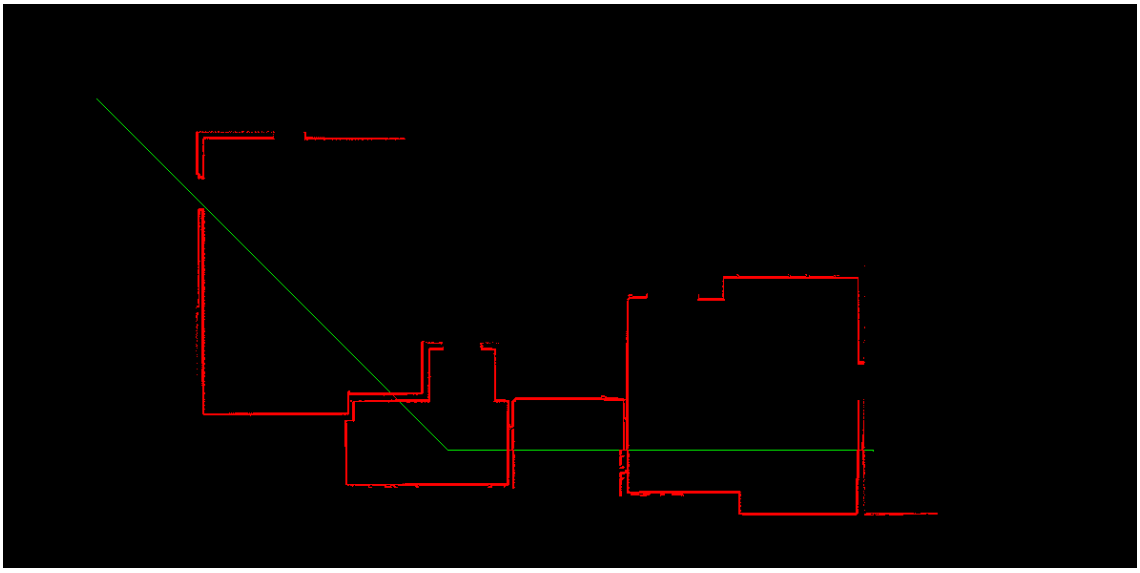

Figure 3: The green line is the path generated from the start position. It passes through the walls as it has not discovered them yet.

4.4.2 Intermediate Nodes

Therefore a solution to both the time and the pathing around walls must be developed. Both issues can be tackled by setting the goal position closer to the drone, as it will have to check fewer pixels before reaching the goal, and if the goal is within range of the sensor then most of the walls should be discovered. Therefore the time it takes to generate a path will be shorter and if the walls are discovered the path should not pass through any of them. However setting the goal position so close to the drone creates a lack of direction for the drone as there is the potential, especially with random goal generation, for the drone to explore a very small area around itself. The solution to this is to generate an intermediate node between the drone and the goal position, that is still close to the drone. Therefore the drone paths the short distance to the intermediate node and sets a new one. This is achieved through finding the unit vector between the two points and multiplying it by the desired amount, to find a point in between.

The vector is found by,

$$v = (x2 - x1, y2 - y1) \tag{1}$$

the magnitude is found by,

$$|v| = \sqrt{(x2^2 - x1^2) + (y2^2 - y1^2)} \tag{2}$$

The unit vector is found by,

$$u = (v/|v|) \tag{3}$$

The intermediate node that is 50 pixels away is found by

$$(x, y) = (x1, y1) + (50 * u) \tag{4}$$

This produces better results with a reduced time to simulate and fewer paths crossing walls. However the simulation time is only increased when the drone does not become stuck in a corner, as discussed in the literary review this is an A* trap, see Appendix 1, and the density of the map can cause the drone to have to scan tens of thousands of pixels to find a solution. This solution may also not be correct as it passes through walls that the drone does not yet see causing further delay to the simulation and does not find the optimal path, as can be seen in figure 4

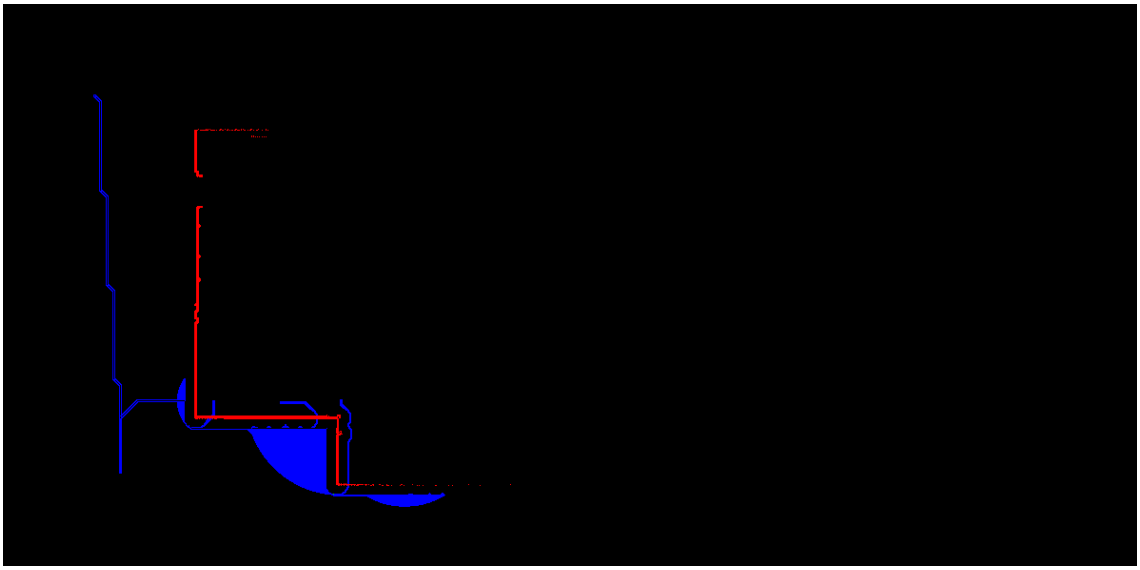

Figure 4: The blue line is the path generated every 100 pixels as the drone reaches the intermediate node between its position and the goal position.

4.4.3 Radial Deflection

The last two solutions clearly show the problem with using A* in a very dense graph like this map. It is that checking tens of thousands of nodes every time the drone wants to go around a wall is impractical and with the map changing as the drone moves this process must happen many times.

The proposed solution to this problem takes inspiration from RRT. RRT would have similar issues to the first A* algorithm where there are too many nodes to check and it would not have a complete view of the map causing the drone to attempt to path through walls. However RRT is similar to the previous solution of intermediate nodes with one noticeable exception, the intermediate nodes are in the line of sight of the drone as seen in Appendix 2. This removes the problem of large search times around walls as the path to the node is trivial due to there being no obstacles in the way.

The proposed solution is therefore a mix of A* and RRT, the solution uses a circle around the drone to set new intermediate nodes by moving around the circle. Due to this, it is called radial deflection. The solution follows the same principles as the intermediate node solution, where between the current position and the goal position of the drone a node is placed that the drone paths to. However, if that

node is on the other side of a wall it is not in the line of sight and the pathing becomes much harder. This means the node must be placed in the line of sight of the drone and still move the drone towards its goal. This is achieved by detecting that a wall is in the way using a similar method to the LIDAR system, then changing the angle of the line to the intermediate node based on a circle drawn around the drone. Once the line generated from this angle does not hit a wall then the new intermediate node is placed there, allowing for a path to be generated quickly, greatly improving the time for each simulation to run and allowing each drone to explore the environment quickly without having to stop for long to generate a new path. As the direction of the deflection dictates how the drones move around the map, half will deflect in the positive direction as seen in figure 5, and half in the negative, therefore two drones starting in the same location and going to the same destination may take different paths depending on whether they deflect in the positive or negative directions.

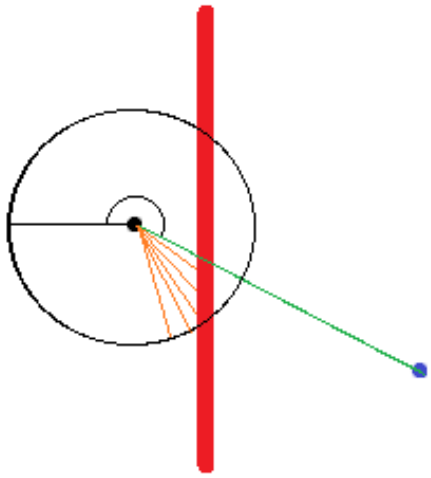

Figure 5: An example of radial deflection. The drone on the left side of the red wall sets a blue intermediate node on the right side of the wall and attempts to path to it. Noticing there is a wall in the way it deflects the line in the positive direction creating multiple orange lines. Once a line is not intercepting a wall or is too close the drone places the intermediate node there.

This can cause a problem where the drone becomes stuck in a corner as it moves back and forth between two points, the solution to this is to detect if the drone deflects by more than 190°and if it does to set a temporary intermediate node 100 pixels away from the drone in an attempt to move it to a better location. The nodes set through redial deflection are only a short distance away from the drone, only 20 pixels. this is to make sure a drone does not overshoot a doorway by deflecting too much. Both of these can be seen in figure 6.

Through multiple iterations of trial and development, the radial deflection technique provides speed of simulation, constant movement of drones and effective path-finding, counteracting all issues presented by just using A* or RRT.

4.5 Drone to Drone Communication

In a similar vein to the ground station, each drone can communicate with any others that are in range. They will share their local maps, disseminating the information throughout the network meaning each drone will carry more information than it has discovered by itself. This has two effects, one being that when a drone returns to the ground station and shares its local environment more of the map is revealed at one time due to the drone having communicated with others. Also if one drone was to be damaged

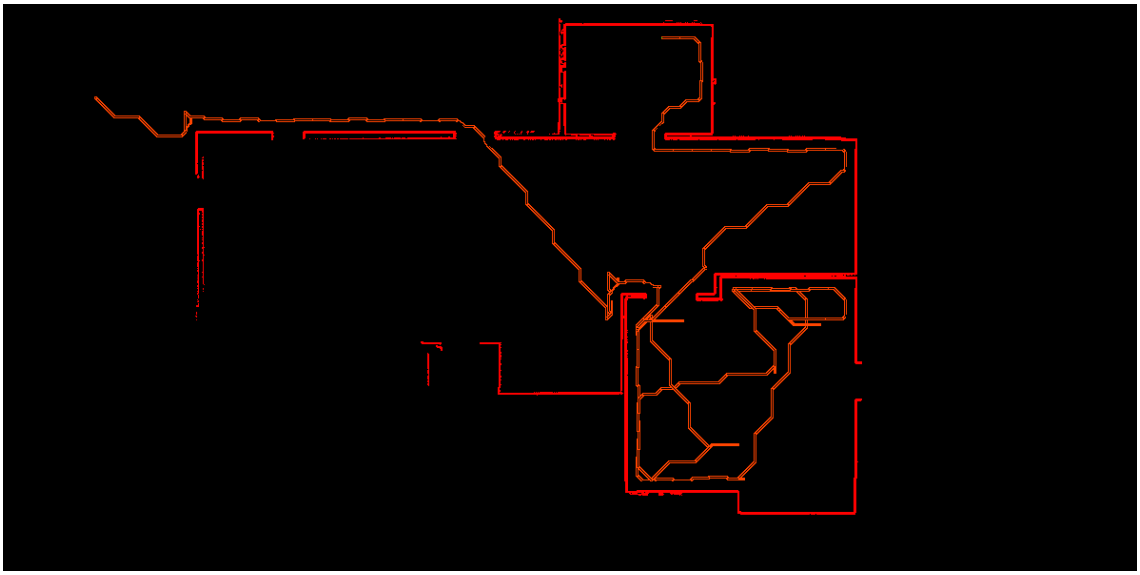

Figure 6: Here a drone can be seen using radial deflection to explore the environment when it is close to a wall the lines between nodes become much shorter to allow it to make tight turns and enter gaps in the wall. the drone can be seen moving a lot through the bottom right room, getting stuck in some corners, due to the repositioning it is eventually able to leave the room.

and could no longer function as it had shared its local environment with others the information and time spent collecting it is not lost, improving the reliability of the swarm as a whole.

Alongside sharing local environments they share mapped chunks, this means that if one drone has already mapped the chunks of another drone it will let it know and therefore those chunks do not need to be searched again. This reduces the time taken to explore the environment by reducing overlapping exploration. A similar effect will happen if one drone detects another is close to its goal position, then marks that goal position as searched and will move on to the next one.

4.6 The Ground Station

All of the drones deploy for a singular entity that in turn directs them and compiles the maps they produce. This is the ground station and it provides an important role in the overall network of the drones.

Having the drones move randomly by themselves works, however the time it takes to map the environment may be improved with better direction. As the environment is known to be a set size it can be split into 100 x 100 pixel chunks by the ground station, a 2D array then describes the environment as these chunks. With these chunks, the ground station can exert finer control over where the drones attempt to map, to map the whole area as fast as possible.

Several mapping styles are implemented within the ground station: with two being vertical liner exploration and horizontal liner exploration where the drones are given a column or row to check before moving to the next one or returning to the ground station. A further style of exploration directs the drones to explore around the outside of the environment first before then moving inside. This allows the drones to outline the building and its entrances before committing to going inside and in this project is called Out In exploration. A version of random exploration is also implemented to work with the chunk system yet functions the same as previous iterations. Finally, a mixed approach is implemented. Where some drones are tasked with in-out exploration, making sure the outside of the building is fully mapped, and some are tasked with random exploration so the inside of the building is quickly being mapped as well. These can be seen in figure 7.

Not only does the ground station direct drone movement, but it also collects their local environments

Figure 7: Top Left: Vertical Exploration. Top Right: Horizontal Exploration. Bottom left: Out In Exploration. Bottom Right: Mixed Exploration.

and combines this data to produce a global map for the operators. It does this by communicating to drones when they are close and adding their local environment to its global environment. The ground station knows which drones are close by querying the environment for drone locations, checking to see if any are in range.

As the ground station tasks the drones with different chunks and collects the global information it becomes the top of a hierarchical structure. This then changes the swarm's network style from a purely decentralised one to a mix, where the drones operate in a decentralised network but sit within a hierarchical one. However, this could provide a distinct point of failure in the system and should be well protected by anyone operating from its position.

4.7 Mapping

The mapping itself is an integral part of the project and is contained within its class, this decision was made as it acts as its distinct subsystem of the drone and it interacts closely with the environment, this abstraction further decouples the drone itself from other areas that it does not need to be part of. The mapping itself, as previously discussed, is a simulation of a 360°LIDAR sensor. This requires the sensor to be able to scan all around the drone, detect a wall and identify where that wall is to be added to the local environment. This is done through a simulated LIDAR system that uses linear interpolation to detect walls in the underlying map and mark them relative to the drone.

Each time a drone senses the environment it scans in a full circle around it, the angels it checks are

determined by using np.linespace between 0 and 2π radians at a set interval. This angle is then used to generate the coordinates of the point the drone is scanning between.

$$x2 = x1 + Range * cos(angel) \tag{5}$$

$$y2 = y1 + Range * sin(angel) \tag{6}$$

Using these coordinates a set number of points between are checked to see if they are a wall.

$$u = i/100 \tag{7}$$

$$x = (x2 * u + x1 * (1 - u)) \tag{8}$$

$$y = (y2 * u + y1 * (1 - u)) \tag{9}$$

For each (x , y) pairing if the colour at that coordinate on the underlying map, that is obtained from the environment class, is black it is a wall and therefore the distance to the point, the angel found and the drone's current position is passed back to the drone class. This process is repeated for every angel and every designated point along the line between the coordinates. Within the drone class, this data is then converted back into coordinates using the same equations as (1) and (2). These coordinates are then stored in the drone's local environment.

The reason the distance angel and position are passed as opposed to just the coordinates is to provide a more accurate representation of the processes that would be happening within the system and to mode the LIDAR as realistically as possible.

The range of the sensor is an important parameter as with an increased range comes a greater awareness of the surroundings allowing the drone to see much further away. As can be seen in figures 8 and 9. This has the effect of making a single drone much more effective, however, it drastically slows the simulation down as it must check many more pixels in the environment.

Figure 8: With a pixel range of only 100 pixels, this drone can not see much around itself

4.8 Cooperative Mapping

The distinct benefit of having a swarm for mapping an environment is the ability to map multiple areas at the same time, therefore the drones must have the ability to map cooperatively, this is even more prevalent if the LIDAR range is low where an increased number of drones gives greater coverage of the environment.

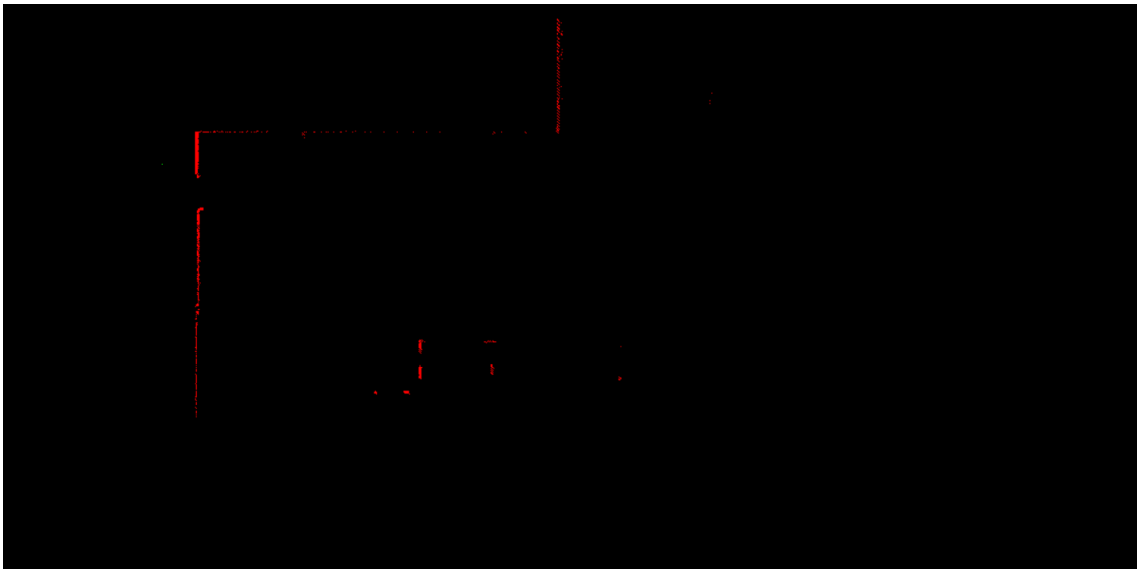

Figure 9: With a pixel range of 300 pixels, this drone can see much more of the environment from a similar location

The drones hold their local environment, using the points they have detected. This is simply a list of tuple coordinates where they have detected a wall. This way of storing the detected walls trivialises the cooperative aspect because for two drones to share data they just append their local environment to the others and remove the duplicates. Therefore each drone can quickly share its information if they are close. As the drones build up their local environments through their scanning and other drones. The overall knowledge for the swarm increases, as this happens the swarm becomes more resilient to information loss from a singular drone failing.

A similar principle is applied to communicating with the ground station where the drone's local environment will be appended to the ground station's global environment and displayed to the user. With multiple drones, the time taken to map the environment is reduced providing more time for the organisation mapping the area to work with. As can be seen n figure **??**, multiple drones can map different areas of the environment simultaneously.

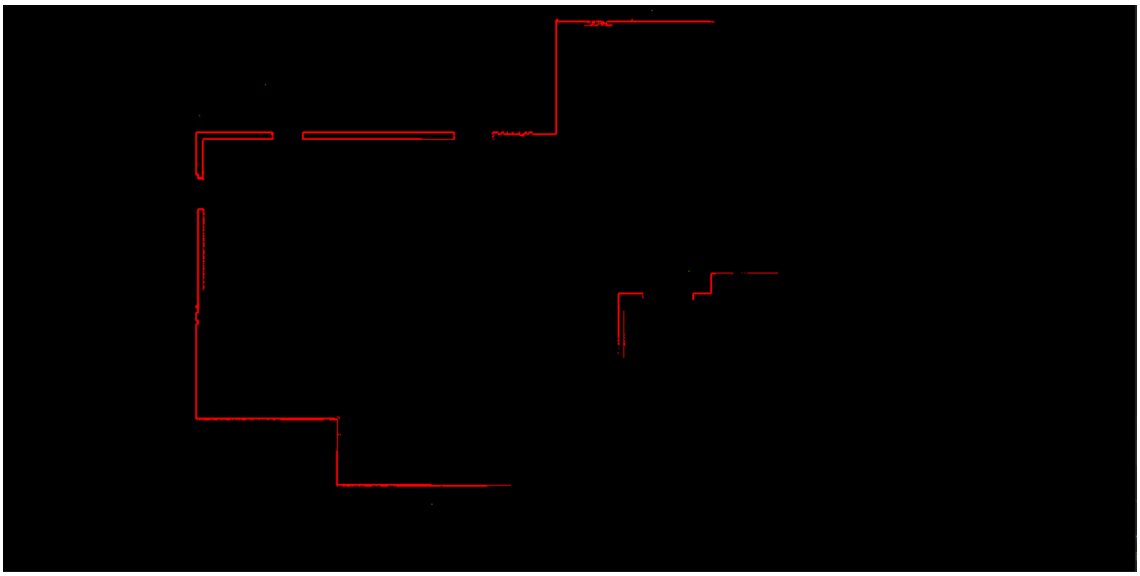

Figure 10: 5 Drones, cooperatively mapping drastically increase the amount of the environment that can be searched at any one time.

4.9 Robustness

With the rest of the simulating in place, the development of a robust network is needed. The swarm achieves this through sharing data throughout the network, where neighbouring drones constantly update and share their local environments. This propagation of information allows data from drones that are not close to the ground station, and therefore cannot communicate with it, to still add their local environment to the global environment through the other drones in the swarm, some of which will be closer to the ground station. This not only improves the time taken for the map to be explored but if the swarm was to lose drones their information would not be completely lost.

A function is implemented where at each simulation time there is a chance that one of the drones will be removed from the environment and therefore cannot move or share data with the other drones, this drone is therefore lost. The system decides when to remove a drone by generating a random number each simulation time between 0 and a set number, if the number is equal to 1 then a drone is removed and the simulation continues. Once all of the drones are removed then the simulation ends and the percentage of the map explored is returned to the user.

4.10 Completion Criteria

The completion criteria are how the system determines when the simulation is complete and enough of the map has been explored. This would be a decision made by the swarm operator on the ground as the global environment is updated by the drones. However, for testing purposes, a limit of 85% needs to be explored for the completion criteria to be satisfied. This is done by taking the original map, flattening it and creating a list of all of the black piles. this will be a list of all of the walls in the environment. Then both this list and the global environment complied by the ground station are counted producing a total for each. Then to find the percentage difference the global environment total is divided by the original map total and multiplied by 100. A decision to limit it to 85% of the map was made as the original map has the full walls included however the global environment from the ground station does not contain the full walls as the drones cannot see inside the walls to determine the colour, therefore the map can never be 100% explored. This effect can be seen in figure **??**.

This process is run every 100 simulation iterations causing a short freeze in the simulation. Once the percentage is equal to or greater than 85% the program stops and produces the number of iterations it took.

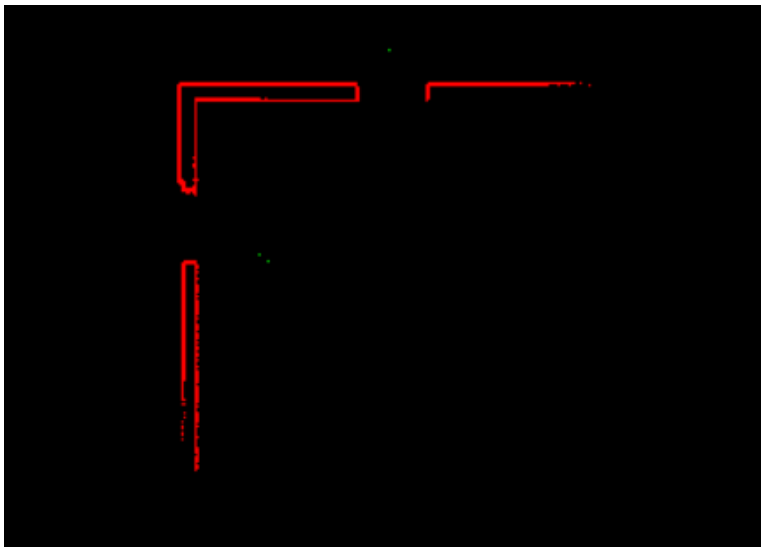

Figure 11: Here the drones have mapped either side of a wall, however, they cannot map the inside, leaving it blank.

5 Results

Several tests can be carried out to determine the effectiveness of the drones and how they work together. These will be based on the average number of iterations it takes to explore 85% of the environment.

5.1 Random vs Directed

These tests are to determine the best style of exploration for different numbers of drones in order to provide a swarm with the best chance of scanning the environment, therefore, the communication distance between the drones and the ground station does not matter when building the global environment. The graph in figure 7 shows a surprising result where the directed exploration styles are worse than the random exploration style, except for the mixed exploration that uses both Random and Out-In exploration.

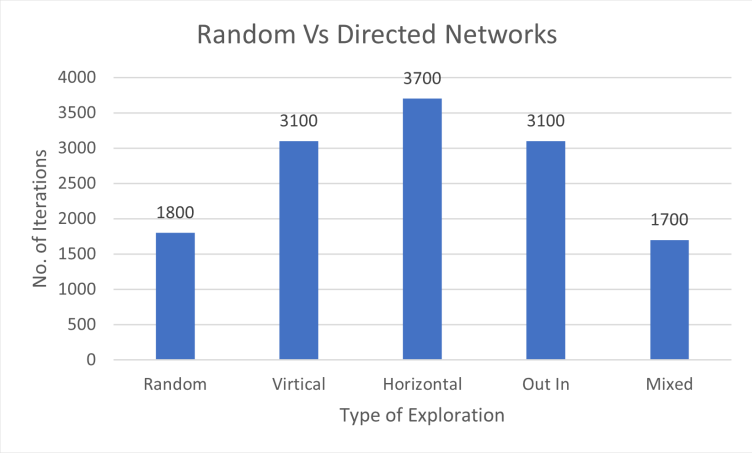

Figure 12: Here the drones have mapped either side of a wall, however, they cannot map the inside, leaving it blank.

These results could be because a random style of exploration has a more even distribution of drones around the environment and as the drones can communicate and share data this would increase the amount of the map in view of the swarm, as opposed to the directed approach where drones can be closer together. The result of the mixed style could be due to both the advantages of the random style and forcing some drones to explore the outside which can sometimes be overlooked in random exploration simulations.

5.2 Communication distance

The modelled communication distance can provide significant changes in the amount of map that can be explored in a given number of iterations.

These tests are conducted using the mixed exploration style with five drones over five tests to find an average.

As seen in figure 13 as the communication distance between the drones increases the fewer iterations it takes to explore 85% of the environment, this is due to the fact that with an increased communication distance the drones do not have to be as close together to share information, allowing the combined map of the environment to spread between the network. This, in turn, speeds up the transfer of the global map back to the ground station increasing the speed of exploration and robustness as the information each drone carries is less unique, therefore if the drone is lost the data it has gathered is more likely to survive.

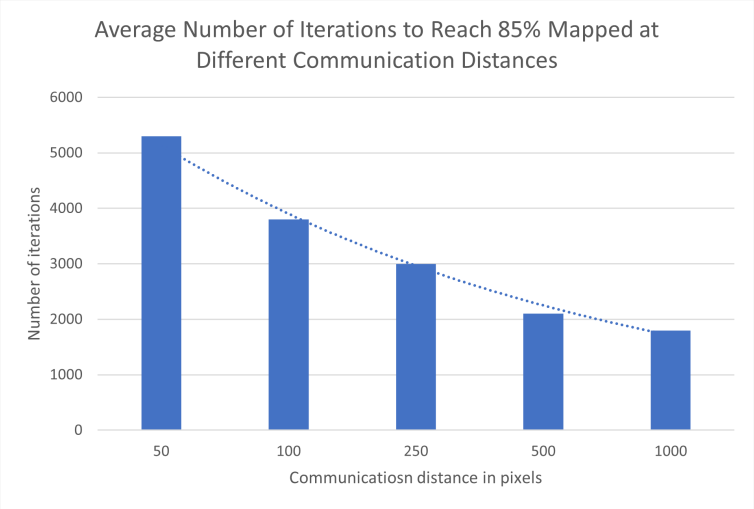

Figure 13: A bar chart showing the number of iterations to explore 85% of the environment with different communication distances.

5.3 Robustness

Testing the robustness of the network involved comparing simulations where the drones can communicate and simulations where they can't. Alongside this throughout the simulation drones could be randomly removed from the environment, as if they had been destroyed, by a percentage chance per simulation time, this is called the attrition rate. The data recorded from previous tests on exploration style and communication distance resulted in these tests being run using 10 drones, on a mixed exploration style with a 250 pixel communication range.

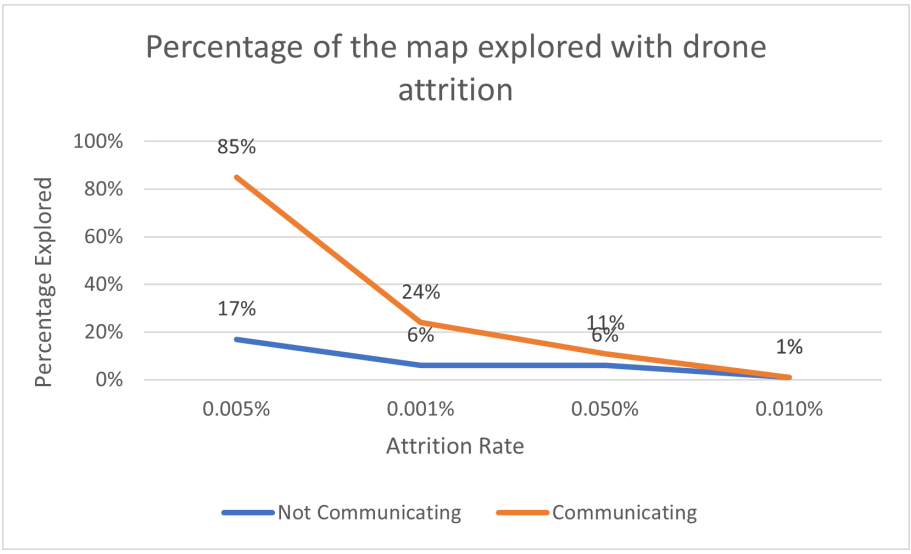

Figure 14: This chart shows the percentage of the map explored before the swarm is destroyed. With the percentage along the x axis being the chance each simulation time that a drone is removed.

As can be seen in figure 14 with a low attrition rate the communicating swarm can explore to the upper limit of 85% and stopped the simulation before all of the drones were lost, whereas the swarm that could not communicate only explored 17% of the environment. This shows that at a low attrition rate a swarm that communicates is much more robust and reliable than one that does not. This trend continues into higher attrition rates but the difference between the percentage explored is much less until it is equal

at 1%. This shows that if the attrition rate is too high it is not worth deploying drones at all as they become ineffective whether they can communicate with each other or not.

This test provides clear evidence of the robustness of the implemented network with a 68% increase in the percentage of the map explored over a not robust network, therefore concluding that the proposed network is an improvement for cooperatively mapping drone swarms.

5.4 Further testing

Here are some proposals for further testing using the simulation.

With the surprising results of the random vs directed exploration testing, the proposed explanation could be further tested by running the same experiments without the drones communicating, this could remove the advantage of even distribution from the random exploration style.

As the robustness test shows, a communicating network is more reliable than one that does not communicate, however, tests to see if this relationship holds when there are more drones in the swarm could result in the percentage explored becoming closer as it takes longer for the whole swarm to be lost.

6 Conclusion

6.1 Aim of the Project and Critical Assessment

This project set out to develop a robust and effective network for drone swarms to cooperatively map an indoor environment. This was achieved through the development of a drone swarm simulation that could map a 2D environment. It also has considered different types of path-finding, developing the blended approach of radial deflection due to the need for simulation speed and pathing accuracy. Alongside this, a robust network was proposed and developed with testing showing that in an environment where drones can be lost or destroyed a robust network is vital to the swarms performance. As the world begins to use drone swarms' in more areas, with ever increasing complexity in automation, these swarms will need to be developed with robustness and reliability in mind for both the effectiveness of the swarm and the safety of the people around it.

In conclusion, the project has successfully provided a system that can simulate a networked drone swarm mapping an indoor environment with several variables available for testing purposes, and show a robust swarm networking approach for operation in dangerous environments.

6.2 Further Steps

After the development of this project, some further improvements could be made in the future or re-developed in a new style. One of these changes could be an improvement to how the drones network with each other by implementing them as instances of microservices on servers. This would allow for a more accurate representation of data transfer using HTTP and REST for communication, further separating the components of the simulation into a more life-like representation. Further to this, using a server would provide more processing power, speeding up the simulation.

Multiprocessing techniques could also be added to the system allowing sets of drones to run on a single-core, splitting the overall workload of the simulation. This would yet further increase the speed of the simulation or allow for the drones to both move and path find at the same time.

Increasing the speed of the simulation would allow for larger swarms and more complex environments to be tested. This testing could in turn show different correlations or different styles of the network to be better or worse as the swarm grows.

As the ground station directs the drones underneath it with hard-coded algorithms it becomes very predictable and suffers from human bias. Using a neural network that would take in setting the number of drones and size of the environment as parameters, could produce a novel solution to directing the swarm in the most effective way, possibly showing a difference between small swarms and large swarms.

One large area of drone research is power usage. This has not been modelled here as it is not a crucial part of the problem, however, to obtain more accurate results in the future the power usage of movement, scanning and communication could be implemented.

The drones in this project are homogeneous, all with the same sensing system and communication range. A future addition to this project could include different types of drones with some that have more powerful communication systems that act as relays and some with different sensors. This mixed approach may change the required style of the network but could reduce the reliability as specific drones carry more importance.

7 Bibliography

References

[1] A. Puttock, A. Cunliffe, K. Anderson and R. E. Brazier, "Aerial photography collected with a multirotor drone reveals impact of eurasian beaver reintroduction on ecosystem structure," *Journal of Unmanned Vehicle Systems*, vol. 3, no. 3, pp. 123–130, 2015.

[2] B. Mishra, D. Garg, P. Narang and V. Mishra, "Drone-surveillance for search and rescue in natural disaster," *Computer Communications*, vol. 156, pp. 1–10, 2020.

[3] A. Milic, A. Ranđelovic and M. Radovanovic, "Use of drones in operations in the urban environment," in *5th International Scientific conference Safety and crisis management–Theory and practise Safety for the future–SecMan*, 2019, pp. 124–130.

[4] S. International, *Shakey the robot*, Mar. 2022. [Online]. Available: `https://www.sri.com/hoi/shakey-the-robot/` (visited on 23rd Nov. 2021).

[5] J. Alkobi, *The evolution of drones: From military to hobby*, Feb. 2022. [Online]. Available: `https://percepto.co/the-evolution-of-drones-from-military-to-hobby-commercial/#:~:text=In%5C%201915%5C%2C%5C%20Nikola%5C%20Tesla%5C%20wrote,the%5C%20Dayton%5C%2DWright%5C%20Airplane%5C%20Company` (visited on 23rd Nov. 2021).

[6] K. Martinez, *The history of drones (drone history timeline from 1849 to 2019)*, May 2020. [Online]. Available: `https://www.dronethusiast.com/history-of-drones/` (visited on 23rd Nov. 2021).

[7] S. Chowdhury, A. Emelogu, M. Marufuzzaman, S. G. Nurre and L. Bian, "Drones for disaster response and relief operations: A continuous approximation model," *International Journal of Production Economics*, vol. 188, pp. 167–184, 2017.

[8] J. J. Roldán-Gómez, E. González-Gironda and A. Barrientos, "A survey on robotic technologies for forest firefighting: Applying drone swarms to improve firefighters' efficiency and safety," *Applied Sciences*, vol. 11, no. 1, p. 363, 2021.

[9] D. Hambling, *Israel used world's first ai-guided combat drone swarm in gaza attacks*, Jul. 2021. [Online]. Available: `https://www.newscientist.com/article/2282656-israel-used-worlds-first-ai-guided-combat-drone-swarm-in-gaza-attacks/` (visited on 24th Nov. 2021).

[10] J. Johnson, "Artificial intelligence, drone swarming and escalation risks in future warfare," *The RUSI Journal*, vol. 165, no. 2, pp. 26–36, 2020.

[11] D. Rojas Viloria, E. L. Solano-Charris, A. Muñoz-Villamizar and J. R. Montoya-Torres, "Unmanned aerial vehicles/drones in vehicle routing problems: A literature review," *International Transactions in Operational Research*, vol. 28, no. 4, pp. 1626–1657, 2021.

[12] O. Borvka, "O jistém problému minimálnım," 1926.

[13] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[14] H. S. Munawar, F. Ullah, A. Heravi, M. J. Thaheem and A. Maqsoom, "Inspecting buildings using drones and computer vision: A machine learning approach to detect cracks and damages," *Drones*, vol. 6, no. 1, p. 5, 2021.

[15] O. Risbøl and L. Gustavsen, "Lidar from drones employed for mapping archaeology–potential, benefits and challenges," *Archaeological Prospection*, vol. 25, no. 4, pp. 329–338, 2018.

[16] R. Ramanathan and J. Redi, "A brief overview of ad hoc networks: Challenges and directions," *IEEE communications Magazine*, vol. 40, no. 5, pp. 20–22, 2002.

[17] M. Soto, P. Nava and L. Alvarado, "Drone formation control system real-time path planning," *Collection of Technical Papers - 2007 AIAA InfoTech at Aerospace Conference*, vol. 1, May 2007. DOI: 10.2514/6.2007-2770.

[18] M. Lupascu, S. Hustiu, A. Burlacu and M. Kloetzer, "Path planning for autonomous drones using 3d rectangular cuboid decomposition," in *2019 23rd International Conference on System Theory, Control and Computing (ICSTCC)*, 2019, pp. 119–124. DOI: 10.1109/ICSTCC.2019.8886091.

[19] S.-G. Cui, H. Wang and L. Yang, "A simulation study of a-star algorithm for robot path planning," in *16th international conference on mechatronics technology*, 2012, pp. 506–510.

[20] H. Wang, Y. Yu and Q. Yuan, "Application of dijkstra algorithm in robot path-planning," in *2011 Second International Conference on Mechanic Automation and Control Engineering*, 2011, pp. 1067–1069. DOI: 10.1109/MACE.2011.5987118.

[21] P. E. Hart, N. J. Nilsson and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. DOI: 10.1109/TSSC.1968.300136.

[22] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[23] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[24] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part i," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006. DOI: 10.1109/MRA.2006.1638022.

[25] D. Zermas, I. Izzat and N. Papanikolopoulos, "Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 5067–5073. DOI: 10.1109/ICRA.2017.7989591.

[26] M. Čikeš, M. Đakulović and I. Petrović, "The path planning algorithms for a mobile robot based on the occupancy grid map of the environment — a comparative study," in *2011 XXIII International Symposium on Information, Communication and Automation Technologies*, 2011, pp. 1–8. DOI: 10.1109/ICAT.2011.6102088.

[27] C. K. Toh, *Ad Hoc Mobile Wireless Networks*, 1st ed. Prentice Hall, Dec. 2001.

[28] O. Shrit, S. Martin, K. Alagha and G. Pujolle, "A new approach to realize drone swarm using ad-hoc network," in *2017 16th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, 2017, pp. 1–5. DOI: 10.1109/MedHocNet.2017.8001645.

[29] Y. Xu and W. Wang, "Topology stability analysis and its application in hierarchical mobile ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 3, pp. 1546–1560, 2009. DOI: 10.1109/TVT.2008.928006.

[30] A. Patel, "Introduction to a*," [Online]. Available: http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html (visited on 17th Nov. 2021).

[31] ". Y. (Jack)". "Motion planning–rapidly-exploring random tree (rrt)." (), [Online]. Available: https://sites.psu.edu/zqy5086/project-1/ (visited on 17th Nov. 2021).

8 Appendix 1

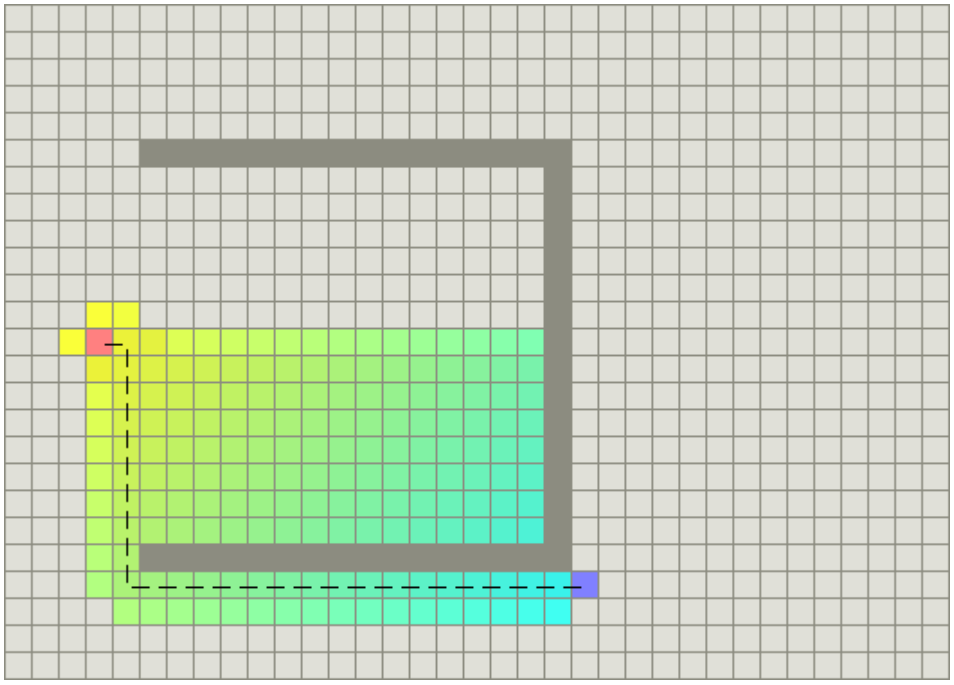

Figure 15: An example of an A* trap, where the algorithm must search a large, and unnecessary portion of the search space due to the corner before the destination [30].

9 Appendix 2

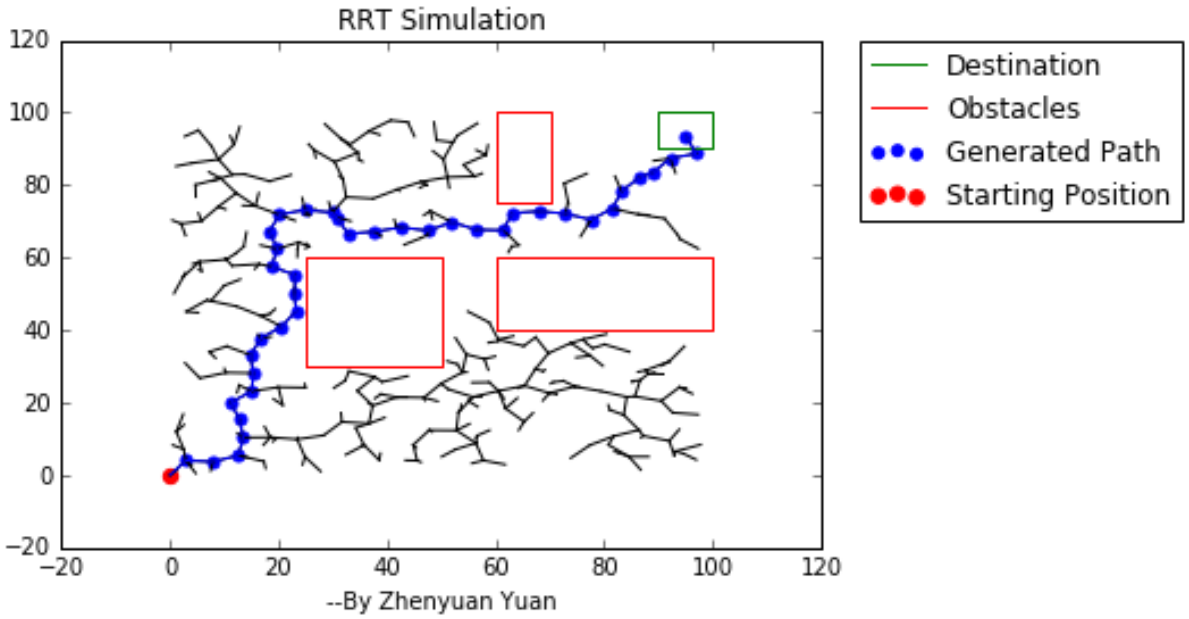

Figure 16: This figure shows a path found by RRT to a destination around some obstacles. It also shows the other branches of the tree as it explored into the unknown space in the environment [31]