Grid-Based Coverage Path Planning with Multiple UAVs for Search and Rescue Applications

by

Welri Botes

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Electrical and Electronic) in the Faculty of Engineering at Stellenbosch University

Supervisor: Dr JAA Engelbrecht
Co-supervisor: Mr JC Schoeman

September 2021

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: 2021/08/30

Abstract

Grid-Based Coverage Path Planning with Multiple UAVs for Search and Rescue Applications

W. Botes

Department of Electrical and Electronic Engineering,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Thesis: MEng (EE)

September 2021

Vibrating a tillage tool is an effective way of reducing the draft force required to pull it through the soil. The degree of draft force reduction is dependent on the combination of operating parameters and soil conditions. It is thus necessary to optimize the vibratory implement for different conditions.

Numerical modelling is more flexible than experimental testing and analytical models, and less costly than experimental testing. The Discrete Element Method (DEM) was specifically developed for granular materials such as soils and can be used to model a vibrating tillage tool for its design and optimization. The goal was thus to evaluate the ability of DEM to model a vibratory subsoiler and to investigate the cause of the draft force reduction.

The DEM model was evaluated against data ...

Uittreksel

TODO: Insert Afrikaans Title Here

("Grid-Based Coverage Path Planning with Multiple UAVs for Search and Rescue Applications")

W. Botes

Departement Elektries en Elektroniese Ingenieurswese,
Universiteit van Stellenbosch,
Privaatsak X1, Matieland 7602, Suid Afrika.

Tesis: MIng (EE)

September 2021

Om ‘n tand implement te vibreer is ‘n effektiewe manier om die trekkrag, wat benodig word om dit deur die grond te trek, te verminder. Die graad van krag vermindering is afhanklik van die kombinasie van werks parameters en die grond toestand. Dus is dit nodig om die vibrerende implement te optimeer vir verskillende omstandighede.

Numeriese modulering is meer buigsaam en goedkoper as eksperimentele opstellings en analitiese modelle. Die Diskrete Element Metode (DEM) was spesifiek vir korrelrige materiaal, soos grond, ontwikkel en kan gebruik word vir die modellering van ‘n vibrerende implement vir die ontwerp en optimering daarvan. Die doel was dus om die vermoë van DEM om ’n vibrerende skeurploeg the modelleer, te evalueer, en om die oorsaak van die krag vermindering te ondersoek.

Die DEM model was geïvalueer teen data ...

Acknowledgements

I would like to express my sincere gratitude to both my supervisors, Dr Japie Engelbrecht and Mr JC Schoeman. Without their guidance and expertise, this project would not have been possible. A further thanks to all those at the Electronic Systems Laboratory who offered advice and support throughout the past two years.

Contents

List of Figures

List of Tables

Nomenclature

Constants

g =	$9.81\,\mathrm{m/s^2}$

Variables

ϕ	Camera sensor to lens angle	[deg]
H	Height from camera lens to ground	[m]
FOV	Field of view (on ground)	[m]
AOV	Camera angle of view	[deg]
f	Camera focal length	[mm]
h_{len}	Camera sensor height	[mm]
w_{len}	Camera sensor width	[mm]
GSD	Ground sampling distance	[cm/px]

Subscripts

x	The x dimension
y	The y dimension

Acronyms

AI Artificial Intelligence

ANFIS Adaptive-Network-Based Fuzzy Interference System

CPP Coverage Path Planning

DARP Divide Areas Algorithm for Optimal Multi-Robot Coverage Path Planning

FOV Field of View

GA Genetic Algorithm

GM-VPC Geodesic-Manhattan Voronoi-Partition-Based Coverage

GSD Ground Sampling Distance

MCPP Multi-Robot Coverage Path Planning

MFC Multi-Robot Forest Coverage

MSTC Multi-Robot Spanning Tree Coverage

PRM Probabilistic Roadmap

PSO Particle Swarm Optimization

RRT Rapidly Exploring Random Trees

SAR Search and Rescue

SLAM Simultaneous Localization and Mapping

STC Spanning Tree Coverage

UAV Unmanned Aerial Vehicle

UGV Unmanned Ground Vehicle

Chapter 1

Introduction

1.1 Background

Unmanned Aerial Vehicle (UAV) are a technology that have gained popularity in various applications [6]. Originally, UAVs required a ground pilot to manoeuvre them, but are becoming an increasingly automated technology. Applications where UAV automation has been used include, but are not limited to, structure inspections[7], smart farming[8], disaster management[9], power line inspections[10], surveillance[11] and wildfire tracking[12].

Most of the research mentioned was done on the premise of using multi-rotor UAVs, quad-rotor vehicles in particular. It is important to note that the term UAVs also encompasses other aircraft types, like single rotor and fixed wing UAVs. Hybrids also exist that contain both rotary-wing and fixed-wing components[6].

Using UAVs poses a considerable advantage in applications like the ones mentioned when compared to Unmanned Ground Vehicles (UGVs). Their capacity to fly over landscapes and around three dimensional structures makes their potential applications increase substantially. Relatively high altitude flying is a key reason why they are well suited to the application suggested in this paper, which is automated coverage path planning for Search and Rescue (SAR) missions.

According to [13], the purpose of motion planning algorithms are determined by the field of research. In control theory, it refers to algorithms that are designed to find trajectories for agents within a non-linear system. This contrasts with the usual focus of control theory on feedback and optimization because the trajectories are usually computed using open-loop methods. Motion planning takes on a subtly different meaning in the world of robotics or artificial intelligence, but the control theory definition is the one that will be used throughout this text.

Coverage Path Planning (CPP) is a variant of the general motion planning problem. Originally, motion planning algorithms were predominantly used to find solutions for start-goal problems[14]. This could mean getting an agent, a UAV for example, from some starting position to some goal position in an environment[15]. Coverage path planning is different from start-goal path planning in that it tries to determine a path for an agent to pass over all points in an environment[14]. It can be used with ground vehicles, for example, to automate field machines for smart farming[16]. Further examples include vacuum cleaning robots, spray painting robots[17], window cleaning robots[18] and automated lawn mowers[19]. For underwater vehicles it can be used for the inspection of difficult to reach underwater structures[20].

Furthermore, there have been developments in the use of UAVs to perform automated search and rescue operations using coverage path planning. Perhaps the most notable example is a project by DroneSAR where they use DJI drones to perform search and rescue tasks. Their implementation includes a mobile application that allows the user to designate a search area manually[1]. Search and rescue operations often span large areas and UAVs fly above most ground obstacles. Therefore, it is realistic that they assume the search environment can be mapped prior to the search operation[21].

DroneSAR uses one drone per search and rescue operation. Once the environment has been designated, the drone performs a back and forth manoeuvre across the area to achieve coverage. The search operation can be halted if the imaging system detects a possible target in the area. The drone can be switched to manual flight mode for closer inspection and the co-ordinates of the target, for example a person in turmoil, can be sent to the search and rescue team. Their system also allows for the manual assignment of way-points to a flight path to bypass the back and forth manoeuvre.[22]

In Figure 1.1, a screenshot of their mobile application is shown. It illustrates the back and forth manoeuvre used to achieve coverage of the designated area.

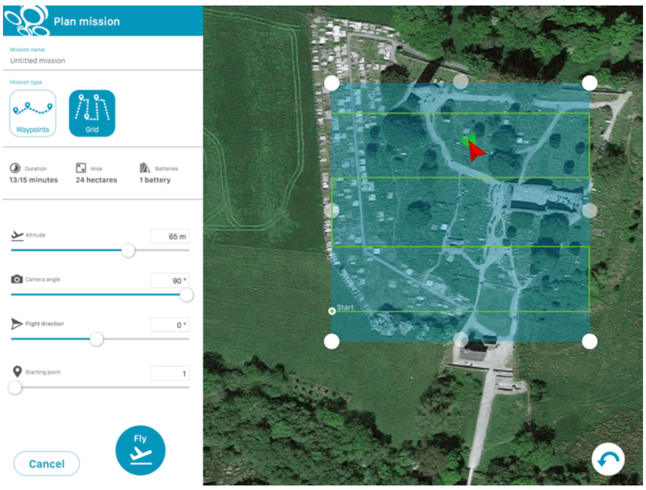


Figure 1.1: DroneSAR Mobile Application Showing Coverage Plan[1]

This paper also looks at coverage path planning for search and rescue, but suggests a multiple UAV approach to the problem. According to [1], when looking for a victim in one square kilometre on land, it takes a five-person rescue team two hours on average to find the victim. DroneSAR found that their drone could do the same job in under 20 minutes. Adding multiple UAVs to cover an area could reduce this time even more, since it would mean more area is covered per unit time. This is important because in a search and rescue operation, time is always of the essence.

This paper also focuses on a grid-based approach to the coverage problem. According to the taxonomy represented in [14], this is referred to as an approximate method. Although one can achieve complete coverage of the grid, the grid itself is not an exact representation of the environment. It does, however, greatly simplify the process of allocating areas to different UAVs, which is a key process for multiple UAV coverage. Physical implementation of this method will not be addressed as part of the scope for this paper.

1.2 Research Aim

The main goal of this research is to develop a coverage path planning algorithm for multiple unmanned aerial vehicles (UAVs) to search an area. The research is intended to be applicable in search and rescue operations using unmanned aerial vehicles to assist.

1.3 Research Objectives

Based on the main aim of this project set out in Section 1.2, a set of research objectives were formulated. These are intended to give a clearer picture of the main research goals and scope of the project. Scope and limitations are further discussed in Section 1.4 and the methodology used to achieve these objectives are detailed in Section 1.6. The research objectives are as follows:

1. Develop a coverage path planning algorithm for an environment that is known a priori and contains static obstacles.

2. Ensure that the final algorithm is an approximately complete solution.

3. Incorporate into the algorithm's functionality an ability to have a changing number of starting UAVs that have random initial positions.

4. Evaluate the algorithm's performance in both randomly generated and mapped, real world environments to ascertain whether or not it is suitable for search and rescue operations.

1.4 Scope and Limitations

1.5 System Overview

1.6 Methodology

1.7 Project Outline

Chapter 2

Literature Review

As mentioned in Section 1.1, CPP is a subset of the general motion planning problem. This chapter will therefore begin with a brief overview of motion planning before discussing CPP as a whole. Literature pertaining to CPP is then discussed in several sections. Firstly, it is addressed in the context of the single robot CPP problem. Several techniques used to achieve coverage using only one robot are summarized.

Following this are three sections dedicated to the Multi-Robot Coverage Path Planning (MCPP) problem. The first two cover distributed and non-distributed offline MCPP respectively. This is followed by a section presenting some online MCPP implementations. Many of the implementations were done with some application in mind, but the last section covers UAVs and how they have been applied to SAR operations in particular.

2.1 Motion Planning

One of the most noteworthy items of literature presented on motion planning is by Lavalle [13]. In this book, a differentiation is made between motion planning and trajectory planning. Motion planning, by their definition, refers to a series of translations and/or rotations required to get an agent from one point to another within some environment. Trajectory planning would then take this plan and find a strategy to execute it within the dynamic constraints of the agent.

Agent is a term from the field of artificial intelligence and is interchangeable with *robot* or *decision maker*. The agent will be what executes the plan once it is determined. Overall, a planning algorithm is used to develop a plan for the agent to execute within an environment. Execution generally refers to a real-world implementation of a plan on some device, for example, a UAV. It can also be performed in simulation. [13]

The type of task that is executed as well as the environment it will be executed in are important for deciding on a motion planning algorithm. The environment may be described as discrete or continuous. Some applications, such as solving a Rubik's cube, can be represented in a discrete manner [13]. Most robotics applications, however, are in a continuous environment, which adds a layer of complexity to problem [23].

Lavalle classifies motion planning problems into discrete and continuous. He discusses point-to-point path planning algorithms such as A* and Dijkstra's Algorithm in the context of discrete path planning. He then classifies continuous problems into two major categories, namely combinatorial and sampling-based methods.

The key difference between these methods is that combinatorial methods explicitly describe the environment, including obstacles, prior to searching and guarantee completeness. Sampling-based methods are generally resolution complete or probabilistically complete, which are more lax notions of completeness. Sampling-based methods sample points in the environment and tend to perform incremental collision avoidance during pathfinding. [13]

The notion of completeness refers to the ability of a planning algorithm to correctly find a solution if one exists, otherwise reporting that there is no solution. Resolution completeness simply guarantees completeness only down to a certain resolution, and probabilistic completeness means that the probability of reporting a correct solution converges to one. [13]

Sampling based methods are often better at dealing with a dynamic environment, whereas combinatorial methods, also known as exact methods, require exact knowledge of the environment beforehand and cannot handle dynamic obstacles without replanning, which is very inefficient. Rapidly Exploring Random Trees (RRT) and Probabilistic Roadmap (PRM) are respective examples of single and multi-query sampling-based methods. Combinatorial methods utilize methods like trapezoidal decompositions and Voronoi diagrams to generate roadmaps. Roadmaps in general can easily be navigated using discrete methods like A*. [13]

Whenever developing a plan, the task could be to move from one point to another, change orientation, or to cover every point within an environment. The task could also involve multiple agents. Optimizing paths in these scenarios can be quite challenging because agents must now not only avoid collisions with obstacles in the environment, but also with one-another, while trying to achieve a certain goal. In the context of path planning with UAVs, the nature of the goal makes the problem fall into different categories, according to the authors of [24].

In a point-to-point problem, if the goal location is the same for all UAVs, it is referred to as a rendezvous task. If they all have different goal locations, it is an allocation task. And lastly, if the goal is not to move from a starting position to a goal, but instead to cover every point in an environment, it is called a coverage task. This classification is highlighted in Figure 2.1.

One final concept to grasp for motion planning, is the difference between online and offline planners. Offline algorithms draw a distinct line between the planning phase and the execution phase. The entire plan is already developed prior to real-world (or simulation based) execution, since the environment is known. Online planners tend to perform planning and execution in tandem. Generally, the environment is sensed as the agent moves and the plan is computed as it goes. The environment is either not known a priori, or is too costly to give as an input for the application. [23]

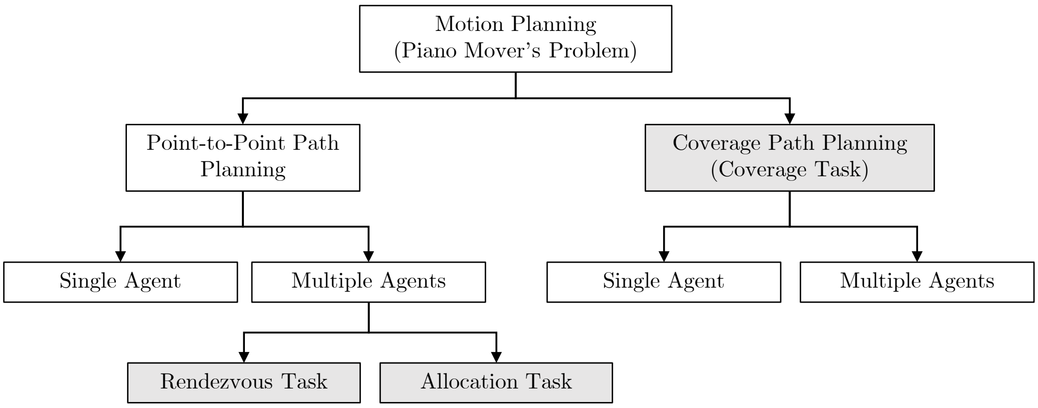


Figure 2.1: Flow diagram showing a breakdown of the different kinds of path planning as part of motion planning.

2.2 Coverage Path Planning

CPP is a subset of the general motion planning problem. The coverage task refers to visiting all points within an environment as opposed to the general start-goal type task [24]. CPP can fall into the same categories as motion planning. It can be classified as discrete or continuous, online or offline, and as a single or multiple agent problem.

A number of surveys have been done to give an overview of the literature available and progress made in the field of CPP. A survey was done in 2001 wherein Choset divides CPP into four categories [14]. In later papers this is known as Choset's taxonomy, and is widely used to describe different types of CPP algorithms.

Choset addresses heuristic and/or randomized approaches and also looks at three type of cellular decompositions, namely approximate, semi-approximate and exact. He continues in briefly addressing the multi robot scenario. The cellular decompositions all rely on simplifying the environment to achieve provably complete coverage.

The approximate methods mean the environment is modelled as a set of cells with equal size and shape, generally a grid. These algorithms can achieve complete coverage of the discrete approximation of the environment, but don't guarantee complete coverage for the actual environment. Exact cellular decompositions divide the environment into polygons and cover these using simple motions e.g. back-and-forth manoeuvres. This can achieve complete coverage of the actual environment, hence the term *exact*. This is further discussed in Section 2.3.1.

In 2013, a survey was done regarding CPP in robotics [21]. It expands on Choset's taxonomy and gives more detail on recent developments in each category. This paper also addresses CPP in three-dimensional scenarios, and briefly looks at CPP where Simultaneous Localization and Mapping (SLAM) is applied due to localization uncertainties. Another recent survey was published in 2019, that once again builds on Choset's taxonomy [6]. They spend more time discussing simple manoeuvres and contrast them with more complex solutions.

Generally multi-robot approaches add a layer of complexity to CPP. The most notable challenge that arises is collision avoidance. Robots need to cooperate to achieve coverage while not only avoiding collisions with obstacles, but also with each other. In 2020, a paper was compiled that specifically deals with cooperative path planning [24]. It surveys path planning with multiple UAVs for the purpose of achieving many different goals, coverage being one of them.

In this chapter, a number of CPP techniques are explored. Methods using a single robot for coverage are discussed in Section 2.3. Sections 2.4 and 2.5 refer to offline methods of MCPP and Section 2.6 looks at online methods. The sections that discuss offline methods are categorised as distributed and non-distributed.

Distributed, for the purpose of this paper, refers to methods where the paths of individual UAVs do not overlap. They are expected to fly in their own isolated sub-regions within an environment. In non-distributed methods,UAVs are free to cross paths. The area division step is not performed and so their paths are simply computed simultaneously, with knowledge of which cells have already been visited. [25]

2.3 Single Robot Coverage Path Planning

Single robot coverage is discussed in some detail here because several of the MCPP problems make use of them. Distributed problems tend to divide an environment into sub-regions that can then be covered using single robot techniques. Several of the other methods simply use the single robot methods and scale them to multiple robot applications. The subsections discuss several existing planning algorithms. This is by no means a comprehensive list of all the algorithms available, but gives a brief overview of varying techniques.

2.3.1 Exact Methods

Combinatorial methods, as described by Lavalle, are also referred to as exact methods [13]. Exact methods for CPP make use of the same geometric principles to divide an area into cells. However, instead of creating a roadmap, an adjacency graph is created and used to move between cells. Each cell is then individually covered, generally using simple manoeuvres [21].

Each cell in the decomposition is a node in the adjacency graph. An exhaustive walk is used to ascertain the sequence in which to visit these nodes to achieve coverage. Simple manoeuvres, such as back-and-forth motions, are then used to cover each cell individually. These algorithms are complete, so will completely cover the environment when possible. [26]

A popular method, that is mentioned in Lavalle's book, is the trapezoidal decomposition. This method decomposes an environment into trapezoids (convex cells) based on the vertices of polygonal obstacles [13].

The trapezoidal and boustrophedon decomposition methods are applicable in two-dimensional coverage problems. They are offline approaches, since the environment must be known a priori, and only operate with polygonal obstacles. The boustrophedon method reduces the number of cells by only looking at vertices where a line can extend both upwards and downward from it. This reduces the final length of the coverage path and makes it more efficient. [21]

A more versatile exact method, that uses Morse functions for the decomposition, is also available [27]. This no longer requires polygonal environments and can in theory be expanded to higher dimensional environments.

2.3.2 Sampling-based Methods

Sampling-based methods have been adapted for coverage path planning. They are more easily scaled to three dimensional environments and are better suited to online or real-time approaches. They also deal with changing environments with dynamic obstacles more easily.

RRT has been adapted to CPP. An example of this is an application involving automated lawn mowing. They used RRT as a local planner in combination with a global planner that uses a spiral motion to cover the points in a map. A solution is however, not guaranteed. Complete coverage is also not necessarily achieved because of the random nature of the paths, but it is considered a real-time approach. [28]

Two other problems, that are both proven to be probabilistically complete, are the watchman rout algorithm and the redundant roadmap algorithm [20]. The watchman method constructs a roadmap, and coverage is achieved using a minimum spanning tree. Roadmap construction is generally done using a PRM [29].

The redundant roadmap method also constructs a roadmap and then conducts point-to-point planning using RRT, where collision avoidance is also implemented [30].

2.3.3 A* and Wavefront Based Coverage

This is a discrete method of planning. As mentioned in Section 2.1, A* is often used in point-to-point path planning. In combinatorial motion planning or multiple-query sampling-based methods such as PRM, roadmaps are generally formed to represent the environment. These roadmaps can then be navigated using A* or another discrete algorithm like Dijkstra's algorithm or a forward search. [13].

A* was built from Dijkstra's algorithm, which is like a forward search that takes cost into account for the priority queue. A* is an extension of this that also predicts the cost to reach the goal using a heuristic.

Dijkstra has also been optimised into what is called a wavefront planner. With this technique, equal cost points are grouped together into "waves" and the algorithm essentially propagates out in waves until it reaches the goal [13]. This wavefront type planning has also been used for CPP, with the goal to minimise rotations and the number of revisited cells [31].

Some authors have taken to extending A* algorithms to CPP as well. Here the environment is generally divided into a grid and the cells can represent obstacles or free space. The starting and end-points are always within free space cells.

One of the earlier interpretations of A* CPP is a combination of the boustrophedon method described in 2.3.1 and A* [32]. This is an online method that constructs boustrophedon regions incrementally and uses A* to move from one region to the next.

In point-to-point path planning, the goal is usually to achieve the shortest path possible and the heuristic function is set up for that purpose. For CPP, the cost function is changed to maximize coverage instead. One such implementation uses A* in a grid-based, offline approach where they try to minimise the amount of cells that get revisited [33]. They use critical waypoints and A* based zigzag motions.

Another implementation uses a heuristic function with the goal of minimising the number of rotations, as was the goal of the wavefront planner mentioned earlier in this section [34]. This is useful, because rotations often consume more energy than straight line motions.

2.3.4 Spanning Tree Coverage

Spanning trees are a appin discrete environments. They can be applied in an offline scenarios or incrementally grown for online applications [35]. A spanning tree is created to reach all nodes in an environment. To make a path efficient, the spanning tree is generally used on nodes that represent the centres of larger cells.

These larger cells are then divided into four smaller cells each, which can then be the cells that are navigated by the agents. To reach each cell, the algorithm simply circumnavigates the tree. When operating in a continuous environment, the environment must be divided into the larger cells. A nice feature that this method provides is that it forms a closed loop path. The agent will always loop back to its starting point. [5]

The spanning trees used are usually minimum spanning trees, constructed using algorithms such as Prim's algorithm. These algorithms can minimise tree weight. This weight can represent distance, or any number of other costs. They can even be used to reduce rotations by encouraging the robot to scan an area along a particular direction. [35]

2.3.5 Artificial Intelligence Methods

One article compared several Artificial Intelligence (AI) techniques for CPP. Four methods were compared, including one that employs a Genetic Algorithm (GA). The four methods are the La Palma attraction, La Palma fuzzy logic, Adaptive-Network-Based Fuzzy Interference System (ANFIS) and Particle Swarm Optimization (PSO) approaches. [25]

All of the methods were implemented in a discretized environment (a grid). In this paper they also generate what they call a *Risk/Occupancy Map*. This is given as an input for their algorithms to encourage searching of certain areas first. For each cell, they generate a potential risk/occupancy value (P). The higher the value, the higher the priority of searching that cell.

They evaluated performance of these algorithms in the context of search and rescue and found that the ANFIS approach gives the best performance for that application. The attraction method works well for environments with maps that don't have a varying P value. Fuzzy logic works well when a big portion of the map has high P values, but has a lower success rate. PSO was shown to not work well at all.

2.4 Distributed Offline MCPP

A well established offline coverage path planning approach involves the divide areas technique. This partitions an area into regions for individual robots to cover. Each robot should then be able to cover its area using one of the individual area coverage techniques mentioned in Section 2.3.

2.4.1 Hexagonal Segmentation

A notable distributed approach uses regular hexagons to segment the area of interest [2]. This implementation is reminiscent of the exact methods often used for single robot coverage path planning mentioned in Section 2.3.1. Exact methods generally divide an area into arbitrarily sized polygons called cells. The robot moves between these cells and covers them using simple motions. In the multi-robot scenario, the area is still divided into cells, but they need to be distributed between the robots evenly for searching. The ideal situation is to assign equal sized areas to each robot so that their path lengths are similar and they can complete their paths at roughly the same time.

Hexagonal cells make it easier to assign cells to robots, seen as they are all of the same size. Hexagons are clustered using the K-means algorithm to ensure a similar number of cells are assigned to each robot. The seeds are synonymous with the robots, therefore once the seed locations are finalized for even cell distribution, the robot initial positions are established. The robots cannot start from any random location, which can be undesirable.

The hexagons that are assigned to a given robot are contiguous and form a sub-region that is then covered using simple manoeuvres. Back and forth motions are generally quite popular. Static obstacles are considered in this implementation, but the smallest obstacle resolution is the size of a hexagon which may not be very representative of the environment. Figure 2.2, taken directly from their paper, illustrates the back and forth manoeuvres used to cover the hexagonal partitions. Black hexagons represent no-fly zones and/or static obstacles. The dark red, green and blue regions represent the sub-regions as they are assigned to the respective robots for coverage.

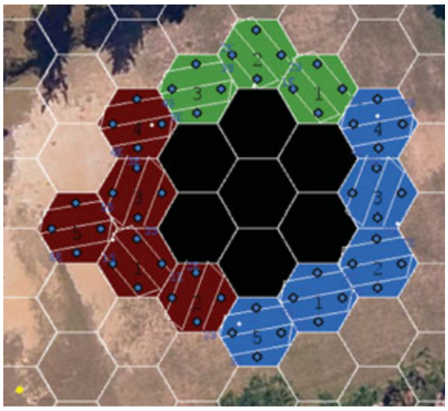


Figure 2.2: Simulation showing coverage of hexagonal partitions with back and forth motions with three robots. [2]

2.4.2 Voronoi Partitioning

In the mathematics field, there are methods of area division to divide a polygon into a number of equal area polygons [36]. Another relevant method that also stems from the field of mathematics, is the Voronoi partition. This assigns regions within an area to seeds based on distance. The idea is that a region assigned to a seed represents all the points where the distance to that seed is shorter than to any other seed.

If the Voronoi partition is applied to the MCPP problem, the seeds become synonymous with robots. This partition works for any number of robots at any starting positions, but unless they are evenly spaced, the areas will not have equal sizes. Distances in these scenarios are usually Euclidean and the boundaries between areas represent the position where the distances from two seeds are equal.

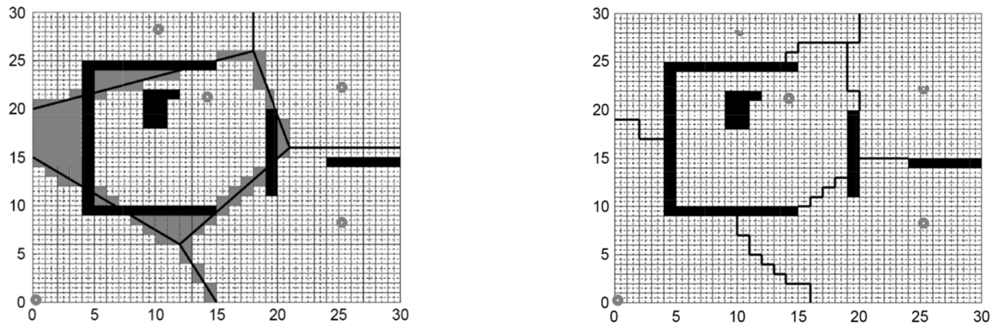
The authors of [3] implement MCPP using Voronoi partitions in discrete space with static obstacles. They use square discretisation of the area and compare several different methods. They investigate geodesic-Manhattan-, Manhattan-, geodesic- and Euclidean-distance-based Voronoi partitions.

The Euclidean-based technique results in what the authors term "non-contiguous sub-regions". This means that cells that are part of a sub-region are not accessible by the robot assigned to them, due to obstacles within that sub-region. They solve this problem by using geodesic distance. This uses Euclidean measurements, but instead of a straight line distance between two cells, it calculates the distance using a collision free path between the two cells.

Another problem arises, due to their use of discrete space. This is that when using Euclidean distances, some cells were partially in two sub-regions instead of fully in one or the other. Their solution is to use Manhattan distances. Ultimately they claim to have solved these problems by using geodesic-Manhattan-based distances to generate the partition. And thus they coined the term Geodesic-Manhattan Voronoi-Partition-Based Coverage (GM-VPC)).

Figure 2.3 shows figures from the paper that show the results of an area division using different distance measures with a Voronoi partition.In both figures the black blocks represent obstacles, the round dots are the robot starting positions and the black lines over the grid represent the partition boundaries.

Figure 2.3a shows the results using Euclidean distances. Here, the grey blocks are areas that would not be covered. This is clearly remedied using the GM-VPC technique in Figure 2.3b. They implemented two different versions of GM-VPC, which utilize respectively an exact and an approximate individual area search technique. They implemented a boustrophedon coverage plan for the exact solution and a spanning tree for the approximate version. Both of these performed better when using geodesic-Manhattan distances.



(a) Euclidean **(b)** Geodesic-Manhattan

Figure 2.3: Figures showing results for the Voronoi partitioning scheme for two different distance measures. [3]

2.4.3 Negotiation Protocol

A negotiation or bargaining protocol refers to a process involving task partitioning. In the context of area division for CPP, the task represents the area to be divided [4]. The authors of [4] present a negotiation model based on Rubinstein's alternate-offers protocol, for the purpose of area division. The focus of their implementation was to develop a distributed algorithm capable of considering robot capabilities. This means that the robots wouldn't have to be homogeneous and can have different flight-time capabilities, manoeuvrability, on-board equipment and so forth [31].

They implemented their algorithm and found that it can achieve near optimum results. It tries to maximise the size of each robot's subdivision of the area (based on its capabilities), while also minimising sub-area overlap. The algorithm also works to avoid static obstacles or no fly zones that are present in the area. Moreover, they proved that it could be applied in a situation where re-planning may be necessary.

A more complete implementation of the algorithm including an individual area search technique was also developed and tested [31]. In this implementation they use a wavefront planner for the individual area coverage path generation. This requires discretisation of the area into cells. In their case, they used rectangles whose size was determined by on-board camera Field of View (FOV). In order for the polygons generated by the negotiation protocol to work effectively, they use a method called Bresenham's line algorithm to approximate the lines that divide the areas in discrete space, so that they pass through the centroids of cells.

The area division achieved sometimes produces non-convex shapes, which the wavefront planner can handle effectively. Their implementation also minimizes energy consumption by minimizing the number of turns and not allowing backtracking. They also have the ability to specify the initial take-off positions of the robots. Distance from the specified take-off point to the starting point for sub-area coverage are considered in the sub-task negotiations. The authors

also mention being able to specify robot landing positions pre-emptively.
One visible drawback in their implementation is that they coverage appears incomplete. The boundaries between areas pass through waypoints (cell centroids), that effectively get excluded from the coverage algorithm and are not covered. Using an exact method to search the individual areas could produce better results. Changing the boundaries to lie on the edges of cells rather than passing through their centroids could also make a difference.

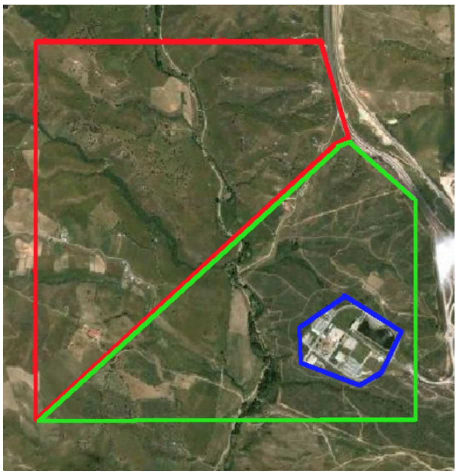


Figure 2.4: Figure showing the resulting area partition using the negotiation protocol. This example is for two robots and includes a no fly zone. [4]

2.4.4 MSTC

Multi-Robot Spanning Tree Coverage (MSTC) is a variant of single robot Spanning Tree Coverage (STC) as presented in Section 2.3.4. The authors of [5] designed the first variants of MSTC. The two variations they suggest are one that allows for backtracking and one that does not. Both variations still utilize a single spanning tree, but simply circumnavigate the tree with multiple robots instead of a single one.
They place a lot of emphasis on robustness and efficiency, in addition to completeness. They demonstrate an algorithm that segments the path around a spanning tree to evenly distribute it among robots. This distribution of robots is however, unrealistic. Their method becomes incredibly inefficient when robots are clustered closely together. This is because a robot simply navigates the path until it reaches the initial position of the next robot on the path.
Figure 2.5 shows the paths that are generated when the robots are evenly distributed along the path that circumnavigates the tree. Blue dots represent the robot initial positions and the spanning tree is shown in red. The second method they suggest remedies this somewhat. It allows for backtracking and

improves the efficiency.
The ideal situation is that all the robots have near equal path lengths, provided they are homogeneous robots. This is not guaranteed with this algorithm when the robots have random starting positions, but allowing for backtracking can improve the results and allow the coverage to be completed in a shorter amount of time.

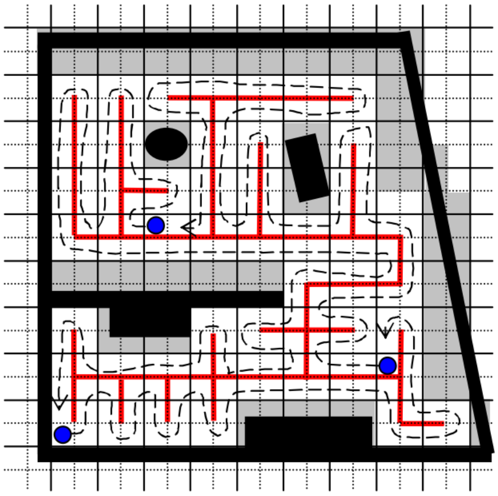


Figure 2.5: MSTC algorithm showing the paths for three robots on an environment grid. [5]

2.4.5 DARP

2.5 Non-Distributed Offline MCPP

2.5.1 MFC

Multi-Robot Forest Coverage (MFC) is a method that was developed by the authors in [37] to improve upon the MSTC method mentioned in Section 2.4.4. Their intent was to construct a tree with the consideration that it will be divided afterwards, unlike what MSTC does. It allows for robot path overlap, which means there is redundant coverage and collision avoidance would need to be considered. However it can handle unique scenarios, where backtracking is unavoidable, quite well.
Their implementation is based on an algorithm presented in [38]. This is an approximation algorithm and they specifically looked at the rooted tree cover scenario. The roots represent the robot initial positions and then a tree is generated for each robot, using the objective to minimize the weight of the maximum weight tree. These trees are each circumnavigated by their robot

(root) to cover the area.
Based on the simulations they ran with MSTC and MFC, they found that MFC generated closer to optimum results and generally achieved coverage in a shorter amount of time. Because there is path overlap in certain scenarios, MFC is not a truly distributed method.

2.5.2 Artificial Intelligence MCPP

Artificial Intelligence based methods have already been mentioned in Section 2.3.5. In this same paper, the application was extended to the multiple robot scenario [25]. They investigate the distributed case, however they don't present their method for dividing the environment into subregions. They also investigate the free formation case for the two and three UAV scenario, and this is of more interest for the purpose of this literature review.

Free formation means that the paths for multiple UAVs are planned in tandem. This means that their paths will potentially cross, implying that collision avoidance would need consideration. Distributed methods explicitly divide an area into sub-regions to be searched by individual UAVs. They will never cross each other's paths, and collision avoidance is not a consideration. This paper does not address collision avoidance in the free formation case. It simply allows the paths to cross.

Three of the methods mentioned in Section 2.3.5 are tested for distributed and free formation flying. PSO was not considered because it performed poorly in the single robot case. In general they found that the La Palma attraction method produces the shortest paths. The ANFIS method comes in close second, and the fuzzy logic approach produces significantly longer paths. Therefore, if coverage time minimisation is important, then the attraction method performs the best.

It should be mentioned that they implemented an occupancy grid, which they generate once. This encourages the algorithms to visit certain regions of the map first. Their performance in this regard may influence one's decision to choose one algorithm over another. This will be discussed further in 2.7.1, as it becomes more relevant in its particular application to SAR.

2.5.3 MCPP Using Linear Programming

2.6 Online MCPP

2.7 UAVs and Search and Rescue

2.7.1 SAR Using Artificial Intelligence

This method has been covered in Section 2.3.5 for its application in the single robot case. It has also been addressed for multiple robots in Section 2.5.2. It should be noted that this paper was specifically developed to investigate these methods for Search and Rescue applications [25].

They developed an risk/occupancy grid for this purpose. Within the environment grid, the generate a risk/occupancy value (P) for each cell. The term is generated by using three different contributions.

The first of these contributions is the terrain factor ($P_{Terrain}$). This is calculated by assessing the likelihood that someone would stay within an area or enter an area and the level of danger they would be in within those areas. This factor has the biggest contribution to the final P-value. There is also an emergency factor ($P_{emergency}$) which represents emergency situations such as fires within an area. Lastly, there is a historical factor ($P_{historical}$) that assesses the likelihood of a historical event occurring again. This is a binary variably.

Using the P-values for all the cells, an risk/occupancy grid is produced that causes the algorithms to favour visiting certain cells first. This grid is not updated recursively. It is only calculated once. In Section 2.5.2, three methods are discussed and it is mentioned that the La Palma attraction approach produces the shortest times.

In a search and rescue operation, short distances are favourable because it reduces time to complete coverage. This means that a target will likely be found faster. However, this paper also has a performance measured called weight. This evaluates the algorithms' abilities to visit high P-value cells first. These are cells that have a high likelihood of containing the target or are high risk zones for the target to be in.

The fuzzy logic approach was found to have the lowest weights of the three methods. However, this is by such a small margin that in the end the ANFIS approach seems to give the best overall performance. It has much shorter distances than the fuzzy logic approach and lower weights than the attraction approach. Therefore, the authors concluded that this approach works best for SAR in both the distributed and free formation case.

2.7.2 DroneSAR

2.7.3 SAR Using Decision Theory

Several works by the same group of authors were presented regarding the use of UAVs for Search and Rescue in the years 2009 and 2010. The authors published their preliminary work regarding coordinated search operations with multiple UAVs [39].

Their setup uses quad rotors searching for a single target in a two dimensional environment. They utilise a downward pointing camera as the sensing device for target detection and onboard GPS for localisation. The environment is divided into a grid and each cell is assigned a probability. This probabilistic map represents a likelihood of the target lying within each cell and forms what is called an *occupancy grid*.

Updating the occupancy grid is done using a recursive Bayesian technique. The assumption of a stationary target means that all the cells are updated for every observation. The occupancy grids are calculated locally on each UAV and are only communicated to others when in communication range. This is referred to as a decoupled approach.

Deciding on a next cell to visit is done by applying a steepest gradient method to the occupancy grid. Their simulation results show that using multiple UAVs, that share information, significantly decreases the time to find the target.

In a separate paper, also by theses authors, they build on their model by including the ability to have multiple observations for one cell and account for changing altitudes of UAVs [40]. Another paper addresses the actual target detection algorithm [41].

For target detection, the once again use a Bayesian estimator and evaluate the probability of target detection at changing altitudes using video data. They found that the sampling rate should be chosen according to the application. For search and rescue it should be chosen so as to minimise false negatives.

Ultimately they conclude that changing altitudes can speed up the search process. They went on to test this strategy online with three different approaches, specifically for Search and Rescue applications. The approaches were designed to deal with information sharing limitations, collision avoidance and uncertainties in the sensor data. The approach that gave the fastest target detection was the Partially Observable Markov Decision Process. [42]

Chapter 3

Environment Representation

This Chapter describes the thought process behind the environment representation. Section 3.1 gives relevant background information and context for the problem, and Section 3.2 describes how this information is used to formulate a discrete environment representation. Lastly, Section 3.3 shows a few practical scenarios using this strategy with different on-board cameras.

3.1 Background

3.1.1 Assumptions

The goal of this paper is to develop a coverage path planning algorithm applicable in a Search and Rescue (SAR) situation, using Unmanned Aerial Vehicles (UAVs). To achieve success in such an application, the first step would be to find a way to represent the environment. Essentially, the context in which the UAVs exist needs to be described in some way for them to navigate it successfully.

The starting assumption is that an offline approach is sufficient. This implies that there is always enough information known about the environment to represent it fully prior to the execution of the path planning algorithm. At the altitude that a UAV flies, it is reasonable to assume that any obstructions, for example power lines or mountains, can be mapped out a priori. Moreover, SAR operations often have fixed geographical bounds in which to search.

Furthermore, it is also assumed that there are no transients in the environment, such as a moving target or dynamic obstacles. Search and rescue operations will rarely have moving elements at the altitude of the UAVs. They are more likely to have fixed no fly zones and static obstacles.

Because they are airborne vehicles,UAVs can execute three dimensional movements, which implies the necessity for a three dimensional representation of the environment. In this paper however, it is assumed that altitude changes are not necessary to search an area. Assuming that there is some form of camera on-board the UAV, a constant altitude will be an advantage. It means that the Ground Sampling Distance (GSD) will remain roughly constant, which is a widely accepted measure of camera accuracy. Assuming a camera is on a UAV pointing down towards the ground, GSD is the distance on the ground as represented by the width of one pixel in an image [43].

Without altitude changes, UAV motions can be represented in two-dimensional space, which greatly simplifies the problem. For offline coverage path planning, it is important to have some demarcated two-dimensional region that requires coverage. The identified region, or environment, can be represented in either a discrete or continuous manner.

In the context of SAR, complete coverage is very important. It will ensure that every possible point in the environment map is covered. In this scenario, it implies that the camera will have viewed all points on the map. Achieving completeness is by no means trivial, but is more achievable in complex environments when using a discrete approach.

3.1.2 Discretisation Using Cameras

Discretising the environment can be done in a few different ways. Assuming a generic UAV with some thermal or visual camera on-board, one has a few options. One can discretise the area based on the UAV size, but a more common practice is to base it off the tool size. In this case, that would be the Field of View (FOV) of the on-board camera. This also makes the process of complete coverage easier, because if the camera is guaranteed to see the entirety of each discrete cell, it is a complete algorithm so long as each cell is visited.

Therefore, to discretise the environment, the FOV needs to be calculated. The camera specifications and UAV altitude will be the determining factors to calculate the FOV. The type of camera and the flight altitude are design decisions and will depend on the GSD necessary to realistically be able to locate the target in a SAR operation.

The diagram in Figure 3.1 shows all the relevant variables needed to calculate the Field of View along one dimension (FOV_x). A similar diagram can be used to calculate the Field of View along the other dimension (FOV_y). The only difference would be the sensor size variable, which changes from the sensor width (w_{len}) to the sensor height (h_{len}). The other variables include the focal length of the camera (f), the height of the lens above ground (H) and the camera's angle of view (AOV). Lastly, there is the variable ϕ which is an angle created due to the sensor being slightly smaller than the diameter of the cone of light projected onto it.

The resulting Field of View will be a rectangle of the same aspect ratio of the camera sensor, provided the camera is pointing directly down and the ground is level. For this application, it is assumed the camera is always pointing downwards. This can be accomplished when the UAV banks by placing the camera on a gimble. The assumption that the ground is level is not entirely reasonable, for example, in a mountainous region. This can be addressed by adding overlap between images to add some redundant coverage, which will be added in Chapter 7. Doing a topographical inspection is beyond the scope of this project and will not be addressed in more detail. The calculations

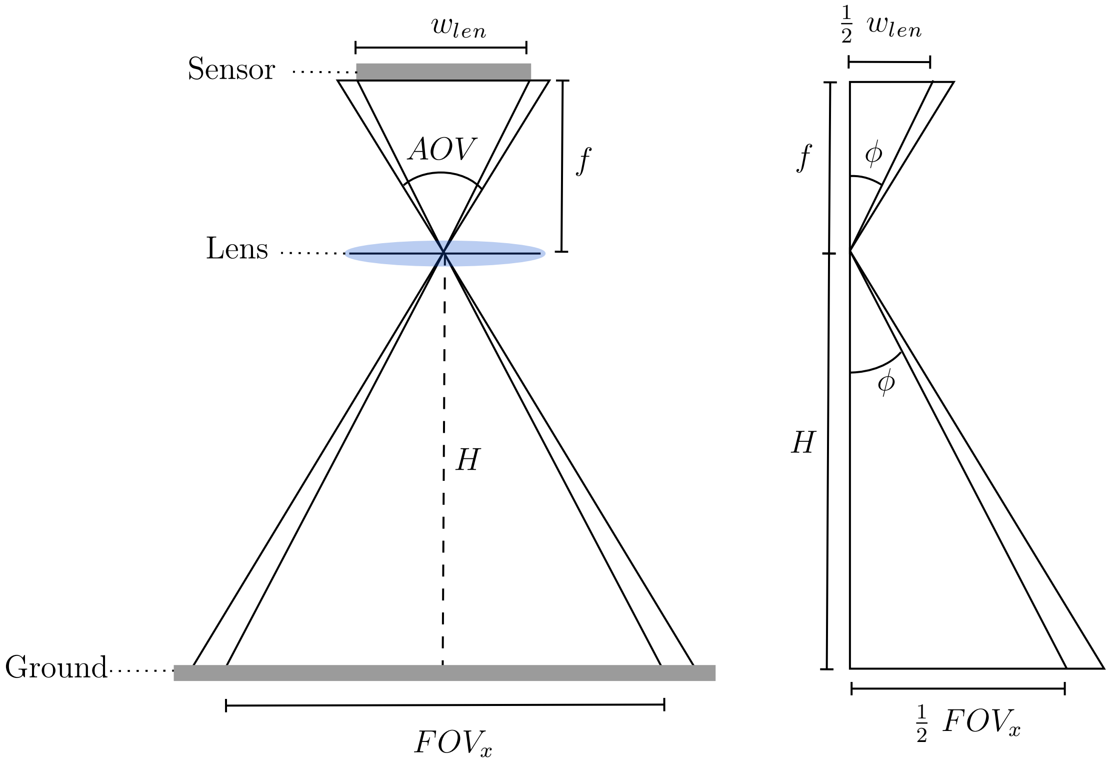


Figure 3.1: Diagram showing relevant variables concerned with calculating the Field of View for a camera.

required to do a discretisation based on the camera FOV will now be discussed. The first equation that is required is the calculation of the angle ϕ which makes use of the small triangle:

$$\begin{aligned} \tan\phi &= \frac{\frac{1}{2}w_{len}}{f} \\ \phi &= \tan^{-1}\left(\frac{w_{len}}{2f}\right) \end{aligned} \tag{3.1}$$

Now that ϕ is known, the bigger triangle is used to calculate FOV_x:

$$\begin{aligned} \frac{FOV_x}{2} &= H \times \tan\phi \\ FOV_x &= 2H \times \tan\left(\tan^{-1}\left(\frac{w_{len}}{2f}\right)\right) \\ FOV_x &= H \times \frac{w_{len}}{f} \end{aligned} \tag{3.2}$$

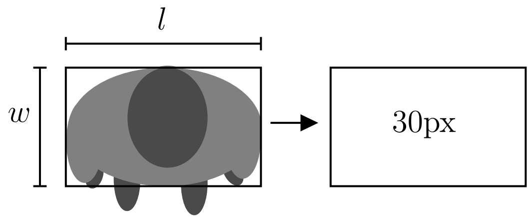


Figure 3.2: Figure showing the rectangular approximation for a human viewed from above for calculation of the GSD

Similarly, FOV_y can be calculated using the other sensor size dimension h_{len}:

$$FOV_y = H \times \frac{h_{len}}{f} \tag{3.3}$$

Both FOV_x and FOV_y will have the same units as H, which is metres. The resolution of the camera is the number of pixels along the image width (px_w) multiplied with the number of pixels along its height (px_h). These can be used to calculate the GSD by dividing either FOV by the pixel value associated with that dimension:

$$\begin{aligned} GSD &= \frac{100FOV_x}{px_w} \\ GSD &= \frac{100H \times w_{len}}{f \times px_w} \end{aligned} \tag{3.4}$$

3.2 Discretisation Methodology

If the size of the environment discretisation is set equal to the rectangular camera field of view, and the type of camera used is known, then a desired GSD can be used to decide on an appropriate flying height. Choosing a GSD would depend on the application, seen as it represents the level of detail that can potentially be detected in an image.

Taking the conservative approach, the assumption is that one is looking for a human being standing upright, viewed from above. To get a good estimate of the space occupied by a human in this orientation, one needs anthropometric data. A survey was done in Europe for people between the ages of 18 and 60 [44]. Among other measurements, they measured chest depth (w) and elbow-to-elbow length (l). These dimensions represent those of an upright human from above and can be seen in Figure 3.2.

To calculate GSD, a minimum number of pixels needed to make a human visible must be chosen. There is an article wherein they developed an image processing algorithm for human detection in a Search and Rescue scenario [45]. In this article they made the decision to put a 30 pixel requirement on human detection. Figure 3.2 shows the rectangular approximation for a human that the 30 pixels should represent.

Using the lower percentile measurements of 170mm chest depth and 390mm elbow-to-elbow length along with the 30 pixel requirement, one gets a GSD of roughly 4.7cm/px. For a child, these values could be even smaller. Therefore, 4cm/px will be used to calculate an appropriate flying height, which is a reasonable value considering that most aerial surveys operate at a GSD of less than 5cm/px [43].
To show how the GSD requirement can be used, a series of cameras are chosen to demonstrate how the discretisation could be determined using the camera specifications and the maximum allowable GSD. Before showing the calculations applied to a specific scenario, they are shown in a more general sense. Firstly, one calculates the maximum allowable height:

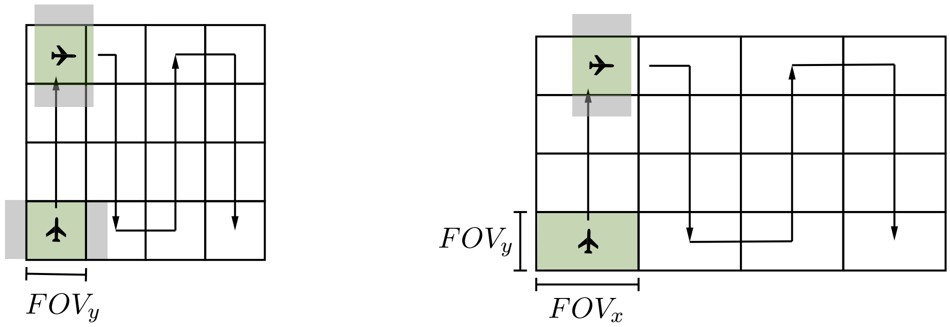
$$H_{max} = \frac{GSD_{max} \times f \times px_w}{100 w_{len}} \tag{3.5}$$

It is desirable to fly as high as possible, because this decreases the coverage time by increasing the camera FOV. Therefore, the assumption is that the height chosen would be the maximum, provided this is within the capabilities of the UAV. Using Equations 3.2 and 3.3, one can then calculate the camera FOV at the height chosen.
One can now discretise the environment based on this field of view. One option is to do a square discretisation. One would make the squares have sides equal to the smaller FOV dimension, which is FOV_y. An example of this discretisation can be seen in Figure 3.3a. This technique will always have cross-track overlap. This means that there will be a percentage of redundant coverage. If a camera has a square FOV then the overlap goes away, but this is uncommon. They tend to have a 4:3 or 3:2 aspect ratio. There is the option of making the squares have side lengths equal to FOV_x, but then complete coverage would not be achievable by simply following the square centroids. This adds a layer of complexity unnecessarily. Setting the square side length equal to l, one can calculate the cross-track overlap using the following equation:

$$\begin{aligned} Provided, \quad & l \leq FOV_y \\ Then, \quad \%Overlap &= \frac{l(FOV_x - l)}{l^2} \times 100 \end{aligned} \tag{3.6}$$

Figure 3.3b shows an alternative technique where the environment is divided into rectangles. With this there is no overlap when moving in the y direction, but there is when moving in the x direction. An environment would be covered faster with this technique provided the y direction is favoured during flight, but no overlap in the y direction is risky. In this scenario, the cross-track overlap over the span of one rectangle when moving in the y direction is zero and for the x direction it can be calculated as follows:

$$\%Overlap = \frac{FOV_x(FOV_x - FOV_y)}{FOV_x - FOV_y} \times 100 \tag{3.7}$$



(a) Square discretisation **(b)** Rectangular discretisation

Figure 3.3: Diagrams showing cross track overlap for camera Field of View for different discretisation techniques.

Both techniques assume sharp 90 degree turns are possible. Some UAVs can make sharp turns, such as multi-rotors, but this often requires deceleration to a hover, which significantly slows coverage time by lowering the average velocity during coverage. Chapter 6 covers a proposed solution for a constant speed scenario. This is considerably more favourable in a scenario where a fixed-wing UAV is used instead of a multi-rotor. Fixed-wing craft are desirable for SAR due to there high endurance. They can generally cover larger areas before needing to refuel.

3.3 Implementation with Different Cameras

Chapter 4

Divide Areas Algorithm

In this project, the distributed method of multi-robot coverage path planning is implemented. Several different algorithms of this type are discussed in Section ??. The algorithm being discussed in this chapter is briefly addressed in Section ??. Section 4.1 addresses Divide Areas Algorithm for Optimal Multi-Robot Coverage Path Planning (DARP) in more detail, and Section 4.2 discusses the implementation of the algorithm with different distance measures. Section 4.3 then goes on to show the implementation of DARP in different practical scenarios.

4.1 Background Theory

Grid-based coverage path planning can be implemented using a number of different methods.

Naturally, achieving the most optimal solution possible would be most desirable. The authors of [46] propose a set of requirements for optimal coverage path planning using a grid-based approach. These fundamental conditions, as they call them, are listed below.

1. Every cell in the environment, that is not classified as an object, must be covered. This is known as complete coverage.

2. Each cell in the environment must only be searched once, and only by one of the robots. This is known as the non-backtracking requirement.

3. Each robot should have as close to an equal amount of cells as possible assigned to it for searching. Their sets of cells should be of roughly the same size.

4. The sets of cells assigned to each robot should be a connected sub-region. This means that when generating a path to search the cells within its set, a robot would not need to traverse that of another to search it's own sub-region.

5. The initial position of each robot should be contained within the set of cells assigned to it. This means that a robot would not need to traverse another robot's sub-region to reach its sub-region for searching.

The authors developed a methodology to achieve these optimal conditions. They called it The DARP. Their solution seeks to divide a known environment containing static obstacles into contiguous sub-regions. These regions are formed based on the robot starting positions so that each robot starts within one of the regions.

The solution is found in an iterative manner and it converges to where the sub-regions are cohesive and roughly the same size. DARP only divides the environment appropriately between robots. A single robot Coverage Path Planning (CPP) algorithm can then be utilized to achieve complete coverage of each sub-region, which translates to complete coverage of the whole environment.

This is a distributed method, which means that each robot travels only within its sub-region. These regions don't overlap, meaning that the robots will never collide, provided they follow their planned path. Therefore, it removes a layer of complexity that generally gets added with multi-robot approaches.

It is also an offline approach to CPP, so the environment is known prior to the planning phase. This phase includes the divide areas algorithm and the sub-region coverage algorithm which is discussed in Chapter 5.

4.1.1 Algorithm Description

The algorithm works on a two-dimensional environment that is divided into discrete cells. Once the discretisation is complete, the algorithm starts by operating similar to a Voronoi partition. It constructs a matrix for each robot with the same dimension as the environment. The authors refer to these as the Evaluation Matrices (E_i). They contain the distances from each cell in the environment to the respective robots. Euclidean distance was used as the distance measure.

4.2 Distance Measure Comparisons

4.3 Implementation with Different Scenarios

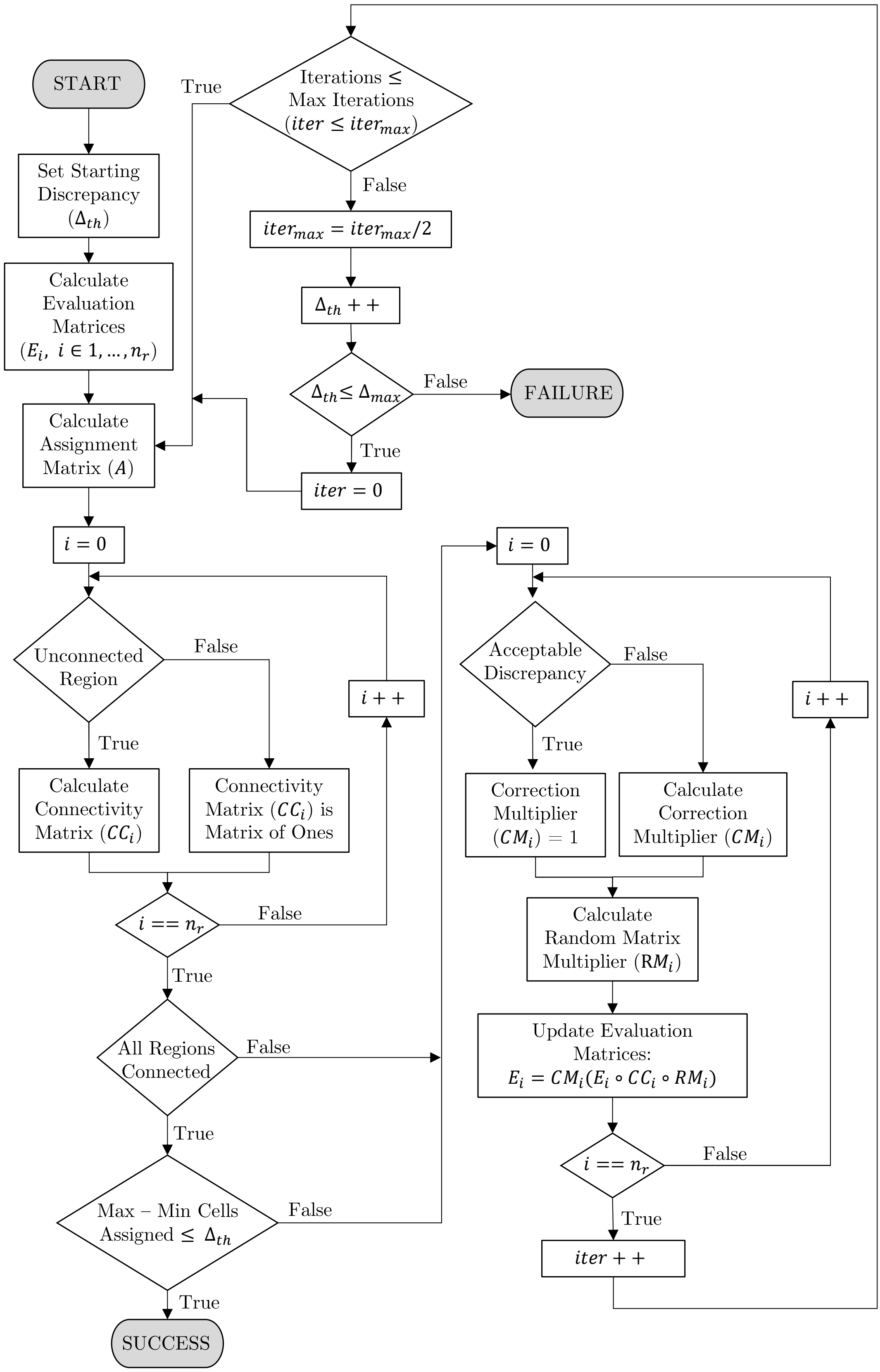


Figure 4.1: Flow diagram representing the logic for DARP.

Chapter 5

Subregion Coverage Technique

5.1 Background

Chapter 6

Dynamic Constraints of a UAV

6.1 Background

Chapter 7

Refuelling Protocol

7.1 Background

Chapter 8

Results

Chapter 9

Conclusions and Future Work

Appendices

Appendix A

Discrete Element Method Theory

A.1 Ball elements

A.1.1 Ball mass and inertia parameters

Consider a volume element $\mathrm{d}V$ with respect to a static base S of an arbitrary solid body with density ρ. The mass of the body is obtained by integrating over the volume of the body,

$$m = \int_{\text{body}} \rho \,\mathrm{d}V \tag{A.1}$$

In figure A.1, a ball with radius R_i and uniform density ρ_i is depicted. The mass of the ball is after integration of equation (A.1)

$$m_i = \tfrac{4}{3}\pi\rho_i\, R_i^3. \tag{A.2}$$

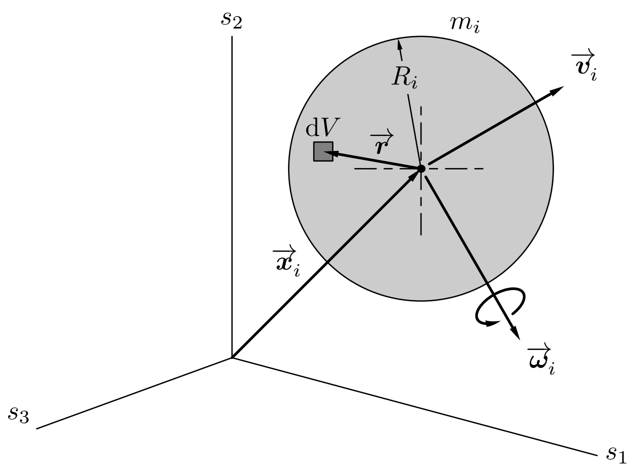


Figure A.1: Ball Element Parameters

List of References

[1] "DJI And DroneSAR Bring Search And Rescue App To First Responders," nov 2016. [Online]. Available: https://www.dji.com/ie/newsroom/news/dji-and-dronesar-bring-search-and-rescue-app-to-first-responders

[2] H. Azpúrua, G. M. Freitas, G. Douglas, and M. F. M. Campos, "Multi-robot coverage path planning using hexagonal segmentation for geophysical surveys," *Robotica*, vol. 36, no. 2018, pp. 1144–1166, 2018.

[3] V. G. Nair and K. R. Guruprasad, "GM-VPC: An Algorithm for Multi-robot Coverage of Known Spaces Using Generalized Voronoi Partition," *Robotica*, vol. 38, no. 5, pp. 845–860, 2020.

[4] C. Rossi, L. Aldama, and A. ententos, "Simultaneous task subdivision and allocation for teams of heterogeneous robots," *Proceedings - IEEE International Conference on Robotics and Automation*, no. May, pp. 946–951, 2009.

[5] N. Hazon and G. A. Kaminka, "Redundancy, efficiency and robustness in multi-robot coverage," *Proceedings - IEEE International Conference on Robotics and Automation*, no. April, pp. 735–741, 2005.

[6] T. M. Cabreira, L. B. Brisolara, and R. Ferreira Paulo, "Survey on coverage path planning with unmanned aerial vehicles," *MDPI*, 2019.

[7] J. A. Guerrero and Y. Bestaoui, "UAV path planning for structure inspection in windy environments," *Journal of Intelligent and Robotic Systems: Theory and Applications*, 2013.

[8] P. Lottes, R. Khanna, J. Pfeifer, R. Siegwart, and C. Stachniss, "UAV-based crop and weed classification for smart farming," *Proceedings - IEEE International Conference on Robotics and Automation*, 2017.

[9] I. Maza, F. Caballero, J. Capitán, J. R. Martínez-De-Dios, and A. Ollero, "Experimental results in multi-UAV coordination for disaster management and civil security applications," *Journal of Intelligent and Robotic Systems: Theory and Applications*, 2011.

[10] W. Chang, G. Yang, J. Yu, Z. Liang, L. Cheng, and C. Zhou, "Development of a power line inspection robot with hybrid operation modes," *IEEE International Conference on Intelligent Robots and Systems*, 2017.

[11] N. Basilico and S. Carpin, "Deploying teams of heterogeneous UAVs in cooperative two-level surveillance missions," *IEEE International Conference on Intelligent Robots and Systems*, 2015.

[12] H. X. Pham, H. M. La, D. Feil-Seifer, and M. Deans, "A distributed control framework for a team of unmanned aerial vehicles for dynamic wildfire tracking," *IEEE International Conference on Intelligent Robots and Systems*, 2017.

[13] S. M. Lavalle, *Planning Algorithms.* Cambridge University Press, 2006.

[14] H. Choset, "Coverage for robotics - A survey of recent results," *Annals of Mathematics and Artificial Intelligence*, 2001.

[15] K. K. M. Lynch and F. C. Park, *Modern robotics : mechanics, planning, and control.* Cambridge University Press, 2017.

[16] I. A. Hameed, "Intelligent coverage path planning for agricultural robots and autonomous machines on three-dimensional terrain," *Journal of Intelligent and Robotic Systems: Theory and Applications*, 2013.

[17] P. N. Atkar, A. Greenfield, D. C. Conner, H. Choset, and A. A. Rizzi, "Uniform coverage of automotive surface patches," *International Journal of Robotics Research*, 2005.

[18] N. Mir-Nasiri, J. Hudyjaya Siswoyo, and M. H. Ali, "Portable Autonomous Window Cleaning Robot," *Procedia Computer Science*, 2018. [Online]. Available: https://doi.org/10.1016/j.procs.2018.07.024

[19] E. M. Arkin, S. P. Fekete, and J. S. Mitchell, "Approximation algorithms for lawn mowing and milling," *Computational Geometry: Theory and Applications*, 1999.

[20] B. Englot and F. S. Hover, "Sampling-based coverage path planning for inspection of complex structures," *ICAPS 2012 - Proceedings of the 22nd International Conference on Automated Planning and Scheduling*, pp. 29–37, 2012.

[21] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, 2013.

[22] O. McGrath, "Are UAVs the future of search and rescue?" *HEMS/SAR*, no. 92, 2018. [Online]. Available: https://www.airmedandrescue.com/latest/long-read/are-uavs-future-search-and-rescue

[23] S. Russell and P. Norvig, *Artificial Intelligence: A modern approach*, 3rd ed. Pearson, 2016, vol. 48.

[24] H. Zhang, B. Xin, L. hua Dou, J. Chen, and K. Hirota, "A review of cooperative path planning of an unmanned aerial vehicle group," *Frontiers of Information Technology and Electronic Engineering*, vol. 21, no. 12, pp. 1671–1694, 2020.

[25] V. S. Juan, M. Santos, and J. M. Andújar, "Intelligent UAV Map Generation and Discrete Path Planning for Search and Rescue Operations," *Hindawi*, vol. 2018, 2018. [Online]. Available: https://www.hindawi.com/journals/complexity/2018/6879419/

[26] H. Choset and P. Pignon, "Coverage Path Planning: The Boustrophedon Cellular Decomposition," *Proceedings of International Conference on Field and Service Robotics*, pp. 203–209, 1997.

[27] H. Choset, E. Acar, A. A. Rizzi, and J. Luntz, "Exact cellular decompositions in terms of critical points of Morse functions," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 3, no. April, pp. 2270–2277, 2000.

[28] N. Nourani-Vatani, M. Bosse, J. Roberts, and M. Dunbabin, "Practical path planning and obstacle avoidance for autonomous mowing," no. January, 2006.

[29] T. Danner and L. E. Kavraki, "Randomized planning for short inspection paths," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2, no. April, pp. 971–976, 2000.

[30] B. Englot and F. Hover, "Planning complex inspection tasks using redundant roadmaps," *Proc. of the International Symposium of Robotics Research (ISRR)*, vol. 100, no. January, 2011.

[31] A. Barrientos, J. Colorado, J. D. Cerro, A. Martinez, C. Rossi, D. Sanz, and J. Valente, "Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots," *Journal of Field Robotics*, vol. 28, no. 5, pp. 667–689, sep 2011.

[32] H. H. Viet, V. H. Dang, M. N. U. Laskar, and T. Chung, "BA: An online complete coverage algorithm for cleaning robots," *Applied Intelligence*, vol. 39, pp. 217–235, 2012.

[33] A. V. Le, V. Prabakaran, V. Sivanantham, and R. E. Mohan, "Modified a-star algorithm for efficient coverage path planning in tetris inspired self-reconfigurable robot with integrated laser sensor," *Sensors (Switzerland)*, vol. 18, no. 8, aug 2018.

[34] S. Dogru and L. Marques, "A*-Based Solution to the Coverage Path Planning Problem Robot 2017," *Advances in Intelligent Systems and Computing*, no. January, 2018.

[35] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of contin uous areas by a mobile robot," *Proceedings - IEEE International Conference on Robotics and Automation*, 1999.

[36] R. Nandakumar and N. Ramana Rao, "Fair partitions of polygons: An elementary introduction," *Proceedings of the Indian Academy of Sciences: Mathematical Sciences*, vol. 122, no. 3, pp. 459–467, 2012.

[37] X. Zheng, S. Jain, S. Koenig, and D. Kempe, "Multi-Robot Forest Coverage *," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005.

[38] G. Even, N. Garg, J. Könemann, R. Ravi, and A. Sinha, "Min-max tree covers of graphs," *International Conference on Approximation Algorithms for Combinatorial Optimization*, 2003.

[39] S. Waharte, N. Trigoni, and S. J. Julier, "Coordinated search with a swarm of UAVs," *2009 6th IEEE Annual Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks Workshops, SECON Workshops 2009*, no. July, 2009.

[40] S. Waharte, A. Symington, and N. Trigoni, "Probabilistic search with agile UAVs," *Proceedings - IEEE International Conference on Robotics and Automation*, no. February 2014, pp. 2840–2845, 2010.

[41] A. Symington, S. Waharte, S. Julier, and N. Trigoni, "Probabilistic target detection by camera-equipped UAVs," *Proceedings - IEEE International Conference on Robotics and Automation*, no. February 2014, pp. 4076–4081, 2010.

[42] S. Waharte and N. Trigoni, "Supporting search and rescue operations with UAVs," *Proceedings - EST 2010 - 2010 International Conference on Emerging Security Technologies, ROBOSEC 2010 - Robots and Security, LAB-RS 2010 - Learning and Adaptive Behavior in Robotic Systems*, no. June, pp. 142–147, 2010.

[43] Propeller Aero, "What is Ground Sample Distance (GSD) and How Does it Affect Your Drone Data ?" 2021. [Online]. Available: https://www.propelleraero.com/blog/ground-sample-distance-gsd-calculate-drone-data/

[44] H. W. Jurgens, I. Matzdorff, and J. Windberg, "International Anthropometric Data for Wok-Place and Machinary Design," vol. 108, pp. 8–9, 1998.

[45] P. Rudol and P. Doherty, "Human body detection and geolocalization for UAV search and rescue missions using color and thermal imagery," *IEEE Aerospace Conference Proceedings*, pp. 1–8, 2008.

[46] A. C. Kapoutsis, S. A. Chatzichristofis, and E. B. Kosmatopoulos, "DARP: Divide Areas Algorithm for Optimal Multi-Robot Coverage Path Planning," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 86, no. 3-4, pp. 663–680, 2017. [Online]. Available: http://dx.doi.org/10.1007/s10846-016-0461-x