COOPERATIVE AREA SURVEILLANCE STRATEGIES USING MULTIPLE UNMANNED SYSTEMS

A Thesis
Presented to
The Academic Faculty

by

Phillip J. Jones

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
May 2009

COOPERATIVE AREA SURVEILLANCE STRATEGIES USING MULTIPLE UNMANNED SYSTEMS

Approved by:

George Vachtsevanos, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Thomas Micheals
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Linda Wills
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Eric Johnson
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Ayanna Howard
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: May 2009

For my Dad. Who knew I would follow in your footsteps?

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. George Vachtsevanos, whose patience and guidance have been invaluable in completing this thesis. Additional thanks go to the other members of my committee, Dr. Ayanna Howard, Dr. Thomas Micheals, Dr. Linda Wills, and Dr. Eric Johnson.

Dr. Graham Drozeski, Dr. Ben Ludington, and Dr. Johan Reimann were co-workers, mentors, and friends. Thanks for keeping the lab environment entertaining.

The research presented here was financially supported by the School of Electrical and Computer Engineering at the Georgia Institute of Technology and DARPA's HURT program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

List of Algorithms

SUMMARY

Recently, the U.S. Department of Defense placed the technological development of intelligence, surveillance, and reconnaissance (ISR) tools at the top of its priority list. Area surveillance that takes place in an urban setting is an ISR tool of special interest. Unmanned aerial vehicles (UAVs) are ideal candidates to perform area surveillance because they are inexpensive and they do not require a human pilot to be aboard. Multiple unmanned systems increase the rate of information flow from the target region and maintain up to date information. The purpose of the research described in this dissertation is to develop and test a system that coordinates multiple UAVs on a wide-area coverage surveillance mission.

In addition to military missions, this research is applicable to a variety of civilian tasks. For example, area coverage algorithms are useful for automating mundane chores such as vacuuming and lawn mowing. In agriculture, these technologies can be applied to automate the delivery of fertilizers and pesticides or to visually inspect crops. An automated system capable of locating stray cattle would save time and money for the ranching industry. Techniques closely related to this research are applicable to tasks as diverse as painting and land-mine removal.

The focus of the research in this document is wide-area surveillance, which includes tasks such as security patrols, forest-fire monitoring, aerial mapping and reconnaissance missions. Information gathered from surveillance is most useful when it is not stale. For example, during security patrols the location of an intruder must be quickly

delivered to capture the intruder and prevent escape. During forest-fire monitoring, it is critical to rapidly pinpoint the location of a newly started forest fire to plan effective strategies that minimize the fire damage. Delayed information in either case could be catastrophic. The system presented here attempts to maximize the revisit rate to ensure timely and accurate surveillance information. One method for increasing the revisit rate of a surveillance task is to assign multiple UAVs to the same surveillance task.

The research presented in this document implements a waypoint generator for multiple aerial vehicles that is especially suited for large area surveillance. The system chooses initial locations for the vehicles and generates a set of balanced sub-trees which cover the region of interest (ROI) for the vehicles. The sub-trees are then optimally combined to form a single minimal tree that spans the entire region. The system transforms the tree path into a series of waypoints suitable for the aerial vehicles. The output of the system is a set of waypoints for each vehicle assigned to the coverage task.

This dissertation details the following contributions:

- A control structure that coordinates multiple aerial vehicles in real-time for an area coverage surveillance task.
- A method for initial placement of multiple aerial vehicles for area surveillance.
- An extension of ground-based planning and control techniques to aerial vehicles.
- A fast waypoint generator for multiple unmanned aerial vehicles to perform surveillance.

- A novel method for constructing and combining disjoint balanced spanning trees.

These contributions are shown to be effective through computer simulation and flight testing at a military base in 29 Palms, California.

CHAPTER I

INTRODUCTION AND MOTIVATION

Unmanned aerial vehicles (UAVs) are well suited for intelligence, surveillance, and reconnaissance (ISR) tasks in both military and civilian domains. Accurate and timely information regarding conditions in a military scenario are crucial for effective planning and utilization of resources. Small UAVs have proven themselves to be valuable in their ability to provide real-time information about the battle space during the conflict in Iraq [37]. Appropriately equipped UAVs are able to provide information from unique vantage points without endangering a human pilot, which makes them particularly desirable for ISR [17, 18]. Much of today's military equipment and organization was designed to counter a single adversary, namely the Soviet Union. Since asymmetric warfare such as drug-trafficking, terrorism, and biological warfare is now considered a more likely threat and the warfare battle space is rapidly shifting from a rural setting into an urban environment, there is a great demand to transform existing military systems to better cope with these threats [12].

One common ISR mission that could be effectively performed by multiple UAVs is that of persistent wide-area surveillance. Suppose that it is desirable to track the location of a target. The target is known to be within a large region of interest (ROI), but the target's specific location is not known. The target must be located before it can be tracked. Small UAVs equipped with appropriate sensors may provide persistent surveillance over the ROI until the target is located, then pass the target's

positional information to a central command center. In turn, the command center relays the information to another team for tracking purposes. If the ROI is large, it may take a significant amount of time to perform the required surveillance, thereby reducing the usefulness and reliability of the collected information. Additional UAVs added to the task may reduce the time required to perform surveillance over the ROI. Unfortunately, at the moment UAVs require significant manpower at a base station to operate. For example, the well-known Predator aircraft made by General Atomics Aeronautical Systems requires two sensor operators in addition to the primary pilot. A full operational team consists of four Predator UAVs and 55 ground personnel to operate the required communication links and maintenance. Additional UAVs require additional manpower, especially to coordinate the flight paths of the individual UAVs. Therefore, there exists a need to develop and implement a system capable of coordinating a surveillance task among a team of UAVs without significant human supervision.

The broad overview of ISR mission intelligence flow is shown in Figure 1. The top of the diagram represents the higher level information flow related to the overall mission objectives, while the lower portions of the diagram represent more specific tasks that combine to achieve the overall mission. The primary focus of this research is closely tied in with the block of this diagram that is labeled "UAV placement and coverage."

A single small UAV offers advantages over a human-piloted vehicle. The UAV can replace several human-piloted vehicles [17, 18] and reduce overall mission costs. A given task can be completed more quickly when multiple UAVs are assigned to the task. While having multiple UAVs does not guarantee improved performance results,

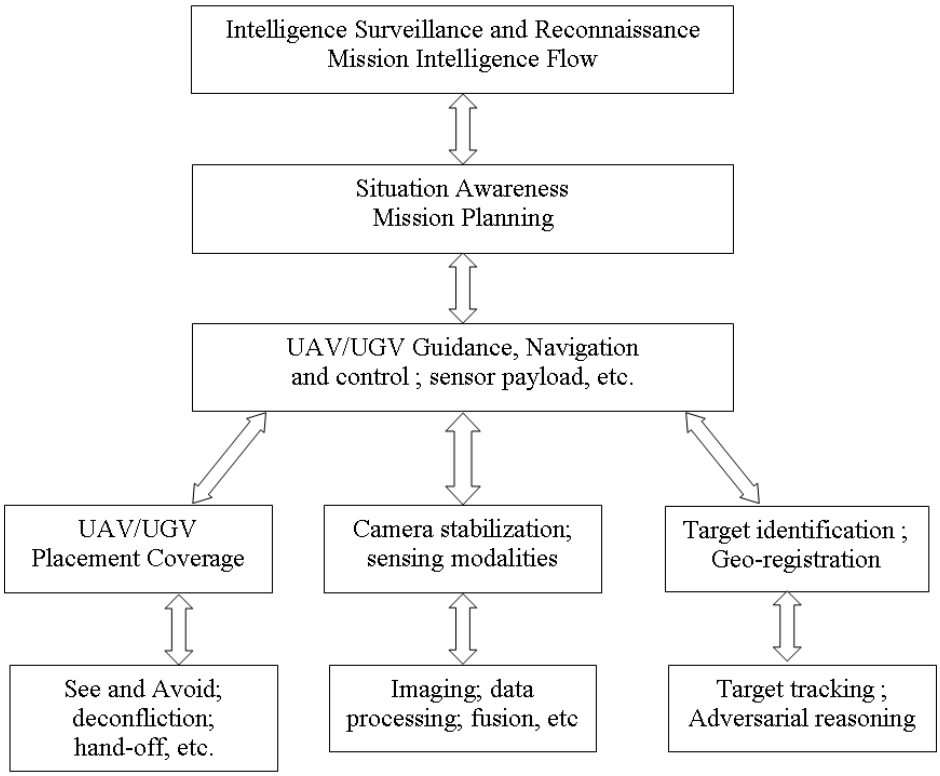

Figure 1: A hierarchical control structure for mission intelligence flow [48].

in general multiple small UAVs retain all the advantages of a single UAV while at the same time increasing the performance and reliability of a given mission [28].

Area surveillance is not limited to military applications. In fact, there are several other interesting real-world applications such as forest fire monitoring, border patrol, aerial mapping, farming and private security.

Area coverage algorithms for ground-based vehicles have been developed for applications such as mine sweeping, lawn mowing and vacuum cleaning [19, 21, 23, 45]. Many of these techniques provide a base from which to build aerial coverage algorithms. However, several distinct challenges exist for aerial coverage that do not directly apply to ground based coverage. For example, a vehicle on the ground is

able to stop and remain stationary while planning its future trajectory. In contrast, fixed-wing aerial vehicles must remain in motion or else they would simply fall out of the sky. Therefore, the path planning algorithm needs to execute in real time. Another challenge is that of vehicle dynamics. While a ground vehicle may make 90-degree turns by rotating in place, an aerial vehicle is constrained by a minimum turning radius, which must be accounted for during path planning.

Further, the small size and light weight of some UAVs affords them the ability to go places a full sized vehicle may not be able to go, but also limits the computational hardware available for a planning routine. Thus, the final system must run in real-time and on modest computational hardware.

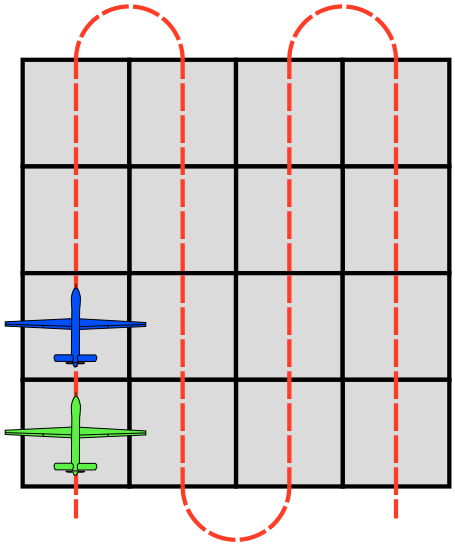

Figure 2: If an additional UAV merely follows the first UAV, there is virtually no improvement in performance.

The purpose of this research is to create a system that generates waypoints for each member in a team of UAVs such that the visited area within a bounded region

is maximized during a fixed interval of time. Initially, the system assumes a homogeneous team of UAVs that have similar capabilities and vehicle dynamics. Additional vehicles should improve the overall performance of the team. As seen in Figure 2, if one UAV merely follows closely behind another UAV the performance increase is virtually non-existent. Therefore, the system must incorporate additional team members in a manner that maximizes the benefit of the additional UAVs.

The revisit rate is the rate at which the entire region is considered to have been viewed by the surveillance team. In addition to the goals of operating in real-time and low complexity algorithms, the revisit rate should also be maximized. In general, a path with few turns requires less time to follow than does a path that includes many turns. Therefore, the first step is to find the principal direction of the ROI to minimize the number of turns required by the UAV to complete the surveillance task. For example in Figure 3, it is faster to cover an ROI that is long but narrow with a path parallel to the long axis rather than a path that crosses back and forth over the narrow section repeatedly.

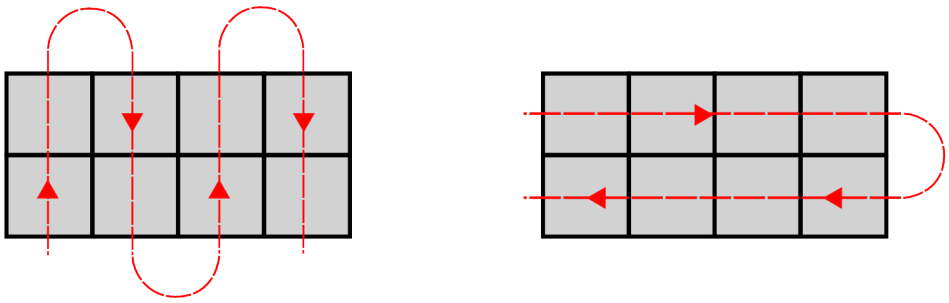

Figure 3: A path parallel to the major axis of the search area is faster than a path that repeatedly crosses the region.

A number of approaches related to region filling have been developed for complete coverage of a region. These techniques typically assume a ground vehicle operating

with a tool, such as a lawn mower or a vacuum cleaner. While these techniques are useful, they are not directly applicable to UAVs because a camera that is rigidly mounted to a fixed-wing aircraft yields a non-linear coverage pattern when the camera footprint is projected to the ground surface [4]. Another difference from previous target searching tasks, region surveillance does not end after being visited only one time but must continue indefinitely [47]. Therefore, it is useful for the UAVs to end in the same location where they began the tour. To accomplish this, path generation is divided into two steps. First, proven techniques are modified to generate an ideal ground path for the centroid of the camera image to follow. Second, the resulting ground path must be transformed into a series of aerial waypoints for the UAV that results in the camera image following the previously calculated ground path. It may be necessary for the camera image to deviate from the ideal path to position the UAV appropriately, especially for turns and cornering. Separating the surveillance task into two distinct steps is advantageous in that one may utilize multiple techniques for coverage and more easily work with a heterogeneous fleet of vehicles when generating the waypoints. The transformation from ground coordinates to air coordinates takes into account vehicle dynamics and camera geometry. Special consideration is given to turning and corners based on a side-mounted, front-mounted, or movable camera system. The resolution requirements of the final image may also be considered in the coverage.

1.1 System Overview

A graphical representation of the cooperative area surveillance system is provided in Figure 4. The operator selects a ROI and assigns UAVs to survey the region.

The system provides an estimated coverage time so that the operator may adjust the number of UAVs assigned to the task. The area is segmented into flyable zones and no-fly zones (NFZs) based on *a priori* information. Next the ROI is segmented into cells before generating the ground paths. In one case, parameterized templates are then selected to cover the region with appropriate paths. In the other case, a graph is created in which each cell from the segmented ROI is represented by a vertex, and connections between cells are represented by edges between the vertices. A tree, consisting of a set of vertices and edges, is created for each UAV assigned to the task and then the trees are connected together such that a single tour about the resulting spanning tree will survey the entire region. Since this tour is currently planned at the ground level, a waypoint generator transforms the path into an aerial path that the UAVs will follow. Video or other sensor information is then relayed from the UAVs to a ground control station to provide real-time information and feedback from the ROI.

1.2 Region Segmentation

Regular and irregular segmentation methods are employed by this system. Regular segmentation involves dividing the ROI into roughly square cells that approximate the area of the entire region. The cell size is determined by the projection of the camera onto the ground. Each cell will be entirely viewable when the camera's image is centered on the cell, as seen in Figure 5. The irregular method sections off a narrow perimeter around the circumference of the ROI and then divides the remaining interior area into rectangular sub-regions. The total number of sub-regions is equal to the total number of UAVs. Restricting UAVs to separate partitions is not necessarily

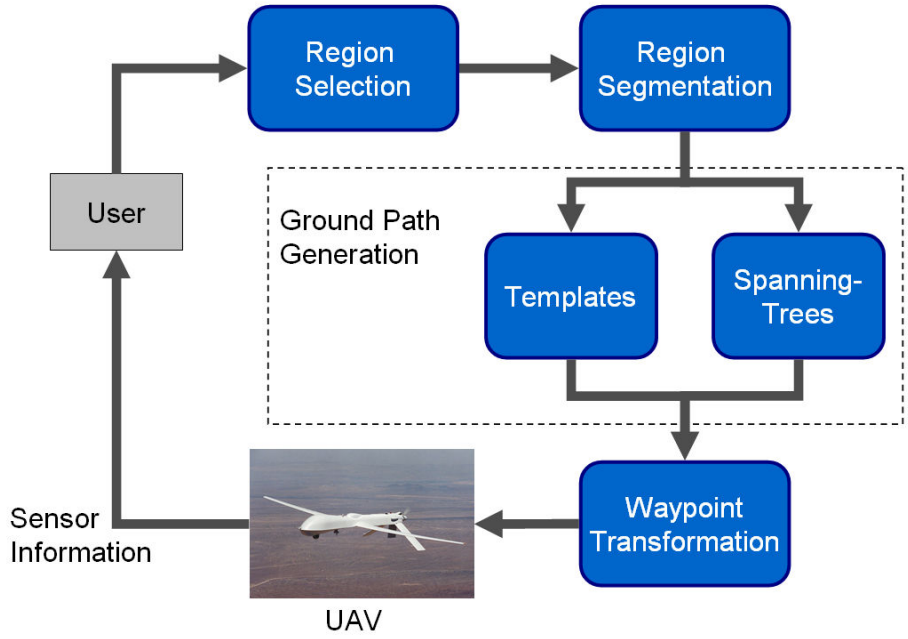

Figure 4: A graphical representation of the cooperative area surveillance system.

optimal but it does offer a convenient and efficient method for task decomposition [3].

1.3 Generating the Ground Path

A path, consisting of a series of waypoints, is generated at ground level. The primary goal in this step is to ensure that every portion of the ROI will be covered in the following surveillance. The ground path represents the ideal path that the camera's image will follow during the coverage routine. These waypoints are calculated either from pre-computed pattern templates or from a spanning tree-based algorithm.

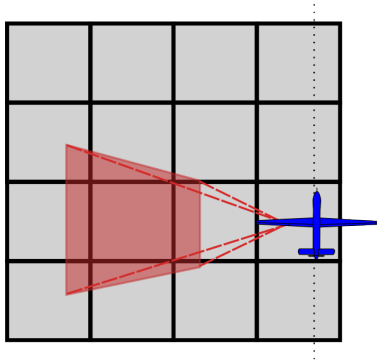

Figure 5: The approximation cells are sized such that a cell can be entirely viewed with one pass of the UAV's camera.

1.3.1 Pattern Templates

The ROI is divided into at least N segments, where N is the number of UAVs assigned to the coverage task. Provided there are at least two UAVs, one segment is always a narrow band that covers the perimeter of the ROI. The remaining interior of the perimeter pattern is then divided into some number of rectangular sections. Parameterized tile patterns are then applied to the segments to create the appropriate ground paths. In this mode, a UAV is assigned to a certain segment of the ROI and UAVs do not transfer from one segment to another. If a UAV happens to fail, then that portion of the ROI will no longer be monitored. However, this method is exceptional at reducing the computational burden.

The primary template utilized in the tiles is a lawnmower type pattern, as picture in Figure 6. There are several cases for this basic shape depending on the entry and exit positions of a UAV into the pattern. Pattern templates are well suited to coverage when the time for planning a route must be minimized.

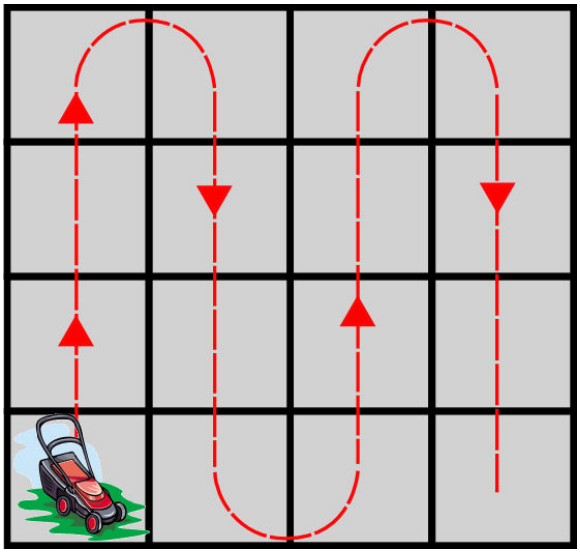

Figure 6: The boustrophedon, or lawnmower pattern, is an efficient path for coverage over a rectangular region.

1.3.2 Spanning Tree Area Coverage (STAC)

A single spanning tree is generated for each vehicle assigned to the surveillance task. Therefore, the ROI is not pre-segmented for the UAVs. Instead, the UAVs are dynamically assigned to smaller portions of the ROI based on a directed spanning tree algorithm that generates the sub-trees. The starting location of each UAV is used as the first vertex of its respective sub-tree. The set of connected vertices is examined, and the vertex that maximizes the minimum distance to the other UAVs' sub-tree is selected. Euclidean distance breaks down in the presence of obstacles so the Manhattan distance metric is used instead. A vertex that may be geographically close to the UAV may be much farther away when obstacles are taken into account. Therefore, when segmenting the ROI into sub-regions the distance metric must follow the path. Eventually, when all flyable zones are allocated, the ROI has been subdivided per UAV. The sub-regions are then joined together to create a single minimum spanning

tree that covers the entire ROI.

1.3.3 Joining the Sub-trees

The individual spanning trees are created in such a way that the overall length is approximately equal for each UAV. Surveillance can begin as soon as the individual trees are complete. In this case, each UAV is assigned to a sub-region within the ROI. The problem with this method is that if a single UAV fails, then the portion of the ROI assigned to that UAV will no longer be monitored. One method to increase the robustness of the system is to bridge the sub-trees together, thus creating a single tour which all the UAVs follow. In the ideal case, the UAVs are spaced exactly equally along this tour which maximizes the revisit rate. To achieve the single continuous tour, the sub-trees are joined together to create a spanning tree that covers the entire ROI. The tour around this tree is then followed by each of the UAVs. For this system, provided at least one UAV remains in operation, the entire region will continue to be monitored.

1.4 Waypoint Transformation

The previously described methods generate a ground tour for the UAVs. The goal is for the centroid of the camera's image to closely follow this ground path. To achieve this goal, the ground waypoints must be transformed into aerial waypoints. The geometry of the camera is taken into account to locate the appropriate aerial waypoint from which to view a given ground waypoint. In addition, the vehicle dynamics are taken into account when making turns. For the case when a vehicle is not able to make a turn with an adequately small radius, it becomes necessary to route the UAV on a longer tour to ensure complete coverage. The transformation

from ground to aerial waypoints must have *at least* one aerial waypoint per ground waypoint and often needs more than one to account for the wider turns. Therefore the number of waypoints will increase in all cases except for the trivial case of a straight line path.

1.5 Organization of this Dissertation

A description of some useful and applicable background technologies is described in Chapter 2. Then, Chapter 3 describes the methods that generate a ground path as used in this system. Chapter 4 provides a description of the method used to automatically transform the ground path into aerial waypoints for the UAV. Chapter 5 is an analysis of the computational complexity of this particular algorithm. Test flight results from a case study program are presented in Chapter 6. Finally, a presentation of simulation results appears in Chapter 7.

CHAPTER II

BACKGROUND

Several background technologies are leveraged to complete an ISR mission. Consider a typical mission scenario in which there are multiple potential targets and multiple mobile sensor agents in the form of UAVs. Figure 7 depicts an example military operations in urban terrain (MOUT) site. In this scenario, the evading targets are at unknown locations within the ROI (the MOUT site). The mission is to locate and track the locations of the targets. Therefore, several UAVs are assigned to surveillance until the targets are found.

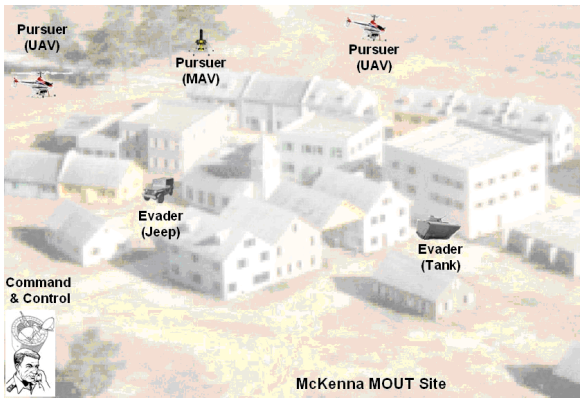

Figure 7: An example of pursuer UAVs and evading targets at a military operations in urban terrain (MOUT) site.

In this example scenario, the UAVs first detect the location of the targets and then transition into a classification and tracking mode to ensure that the targets' locations are not lost. For example, the weighted cooperative multi-robot observation of multiple moving targets (W-CMOMMT) as described in [31] and [30] could take

over in the tracking duties. If the targets are assumed to be intelligent and are aware of the pursuing UAVs, they will likely perform evasive maneuvers in an attempt to escape. Adversarial reasoning and pursuit and evasion games are utilized when the target attempts to avoid surveillance.

The overview of this ISR system is depicted in Figure 8. The upper portion of the graphic describes the higher level reasoning and mission objectives, while the lower portion of the graph is related to the low-level vehicle controls. The middle portion is related to situational awareness and decision making processes necessary to complete the overall objective. The research presented here is motivated by situation awareness, specifically area coverage and sensor placement.

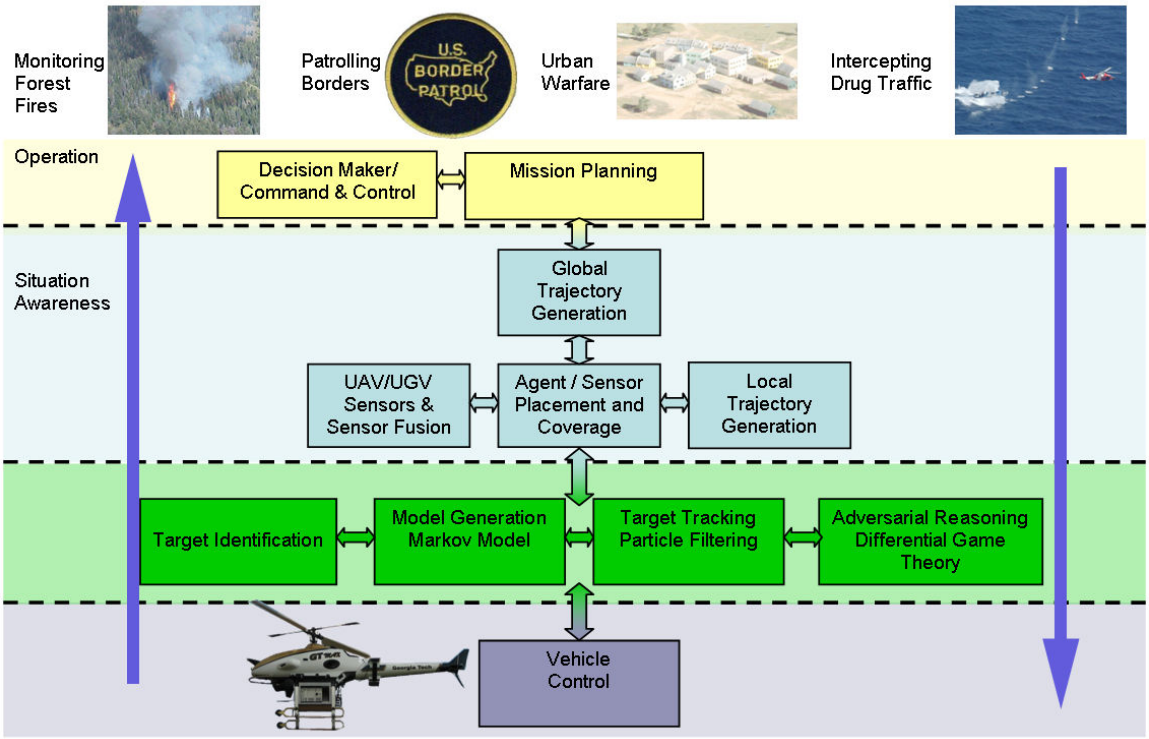

Figure 8: An overview for an ISR system architecture.

This research stems from previous research in the single and multi-vehicle coverage problem. The building blocks and previous research upon which this coverage planner is built are reviewed in the following sections.

2.1 Single Vehicle Complete Coverage

As research in mobile robotics developed, several practical tasks ranging from lawn mowing and vacuuming to mine sweeping were considered. This class of problem can be solved with a coverage planner. The early research in developing coverage algorithms focused on individual ground-based mobile robots. For example, Huang et al. developed a graph-theoretic approach to generate a best path for region filling [35]. In this work, *best* was defined as the shortest path distance with the least energy consumption. Further refinements on the single mobile robot approach concentrated primarily on achieving completeness of coverage and expending minimal energy.

For minimal energy, Huang divides the total energy cost into straight line operation and turning operations. The total energy is represented by

$$E = E_L + E_T \tag{1}$$

where E is the total energy consumed, E_L is the energy consumed in straight-line operation, and E_T is the energy consumed in turning operation. Examining only the straight-line portion of Equation 1, it is found that

$$E_L = F_L \frac{A}{W} \tag{2}$$

where F_L is the force expended to move along the straight line path, A is the area of a region, and W is the width of one strip. The energy consumed by turning operations

is represented by

$$E_T = D \cdot \frac{E_{oT}}{W} \tag{3}$$

where D is the span of a region in a direction and E_{oT} is the energy consumed at each turn. Combining these equations yields the total energy for operation:

$$E = F_L \frac{A}{W} + D \cdot \frac{E_{oT}}{W} \tag{4}$$

Since A, W, F_L, and E_{oT} may be considered constants, the span, D, must be minimized in order to minimize the total energy consumed. The minimum span direction is perpendicular to the principal direction axis. Therefore, utilizing the principal direction reduces the energy and time consumed in the surveillance mission by minimizing the number of turns required in the coverage path.

Focusing on complete coverage, Gabriely developed three planning algorithms that utilize spanning trees [21]. The three versions of the spanning tree coverage (STC) algorithm are the on-line STC, the off-line STC, and the ant-like STC. The on-line version allows for lack of knowledge of the work area and the spanning tree is constructed as the mobile robot moves about the work area. The off-line version assumes that the mobile robot has *a priori* knowledge of the work area. Unfortunately, this algorithm "lacks the means to affect the shape of the spanning tree being constructed" [21]. One key feature of the research presented in this document is that the trees are dynamically shaped. The ant-like STC utilizes a marking device (such as a chemical residue, or physical marker) to indicate that a location has been previously visited. The marking technique is employed by ants in nature but is still not practical for

mobile robots and clearly not well suited for UAVs. The basic STC algorithm attempts to embed a Hamiltonian cycle, which is a path that visits every vertex exactly once [42], having the largest number of cells in the given work area. All three of the proposed algorithms provide complete coverage of the work area and the coverage is considered optimal in that there is no repetitive coverage.

Arkin and Refeal suggest that the coverage path problem is related to the covering salesman problem, which is itself a variant of the traveling salesman problem [6]. Instead of visiting each city, the agent must visit a neighborhood within each city that minimizes the travel length for that agent. In [7], the Arkin et al. provide algorithms for the basic forms of the lawn mowing and milling problems. Arkin also demonstrates that these problems require nondeterministic polynomial time to solve (NP-hard) in general.

Choset and Pignon address the traveling salesman nature of the problem and describe an off-line planning algorithm that acts on a polygonal world. The robot's free space is broken down, or decomposed, into cells in their boustrophedon (ox-like) cellular decomposition. The algorithm is an exact decomposition approach in which each sub-cell is covered by a simple line sweep [13]. Therefore, complete coverage reduces to treating each sub-cell as a node in a graph and finding the path through the adjacency graph that visits each node at least one time. Hert describes a similar system in [32]. Hert's system also uses a line sweep algorithm on sub-regions and a traveling salesman heuristic to connect the sub-regions. However, Hert's system operates on non-polygonal worlds.

Choset later provides a survey of coverage algorithms in [14]. The existing approaches are divided into three major categories based on how the entire work area is

divided into cells. The three approaches are labeled *approximate*, *semi-approximate*, and *exact* cellular decompositions. These types of decompositions are illustrated in Figure 9, Figure 10, and Figure 11, respectively. For approximate decompositions,

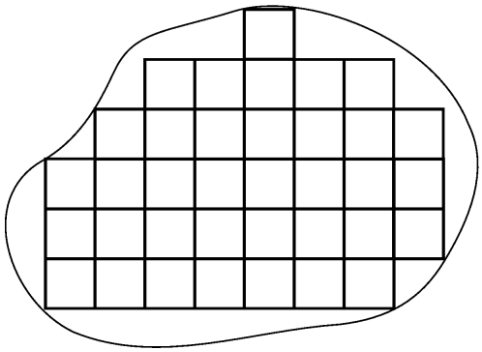

Figure 9: An approximate decomposition of the ROI.

the cells are all the same size and shape; thus the work area is approximated by the cells. Any cells that partially cover an obstacle or lie partially outside the work area boundary are discarded. In semi-approximate decomposition, cells are fixed in

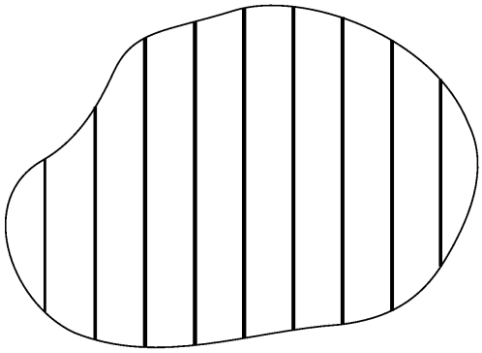

Figure 10: A semi-approximate decomposition of the ROI.

one dimension (width, for example) but can have any shape in the other dimension (height, for example). The semi-approximate decomposition reduces the amount of inaccessible space while maintaining low memory requirements. For exact cellular de-

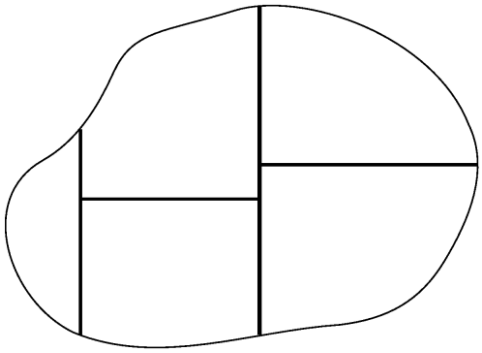

Figure 11: An exact decomposition of the ROI.

composition, the environment is divided into a set of non-intersecting regions whose union fills the entire target environment. One popular technique that falls into this category is the trapezoidal decomposition (also known as the slab method). The boustrophedon decomposition described by Choset is a generalization of the trapezoidal decomposition. Choset also proposes that "path planning is the bottleneck for applications such as mine sweeping, oceanographic mapping, and painting" [14].

Huang [35] adapts the boustrophedon approach to achieve optimal coverage by optimizing the total number of turns required to cover all the sub-regions. Huang shows that the optimal line sweep must be parallel to an edge of the boundary and the obstacles. Later research in [34] describes a method of sub-dividing the region of interest into subregions and then applying a planar sweep to each subregion. Again, it is noted that reducing the number of turns improves the coverage time. Unfortunately, the algorithm presented by Huang has exponential complexity and therefore is not suited to small UAVs.

Another concept for complete coverage involves critical points. Garcia [23] describes an enhancement to Choset's coverage method that utilizes exact cellular decomposition and critical points. In this work, the vehicle is assumed to occupy a

finite area rather than being modeled as a point and assumes that the robot is able to determine its own position within the environment. The approach presented addresses the inability of previous algorithms to handle unstructured environments and results in a single vehicle coverage algorithm that is more complete while retaining a performance metric that is as good as the previous methods.

2.2 Multi-Vehicle Improvements

As the algorithms for complete coverage utilizing a single vehicle improved, this research was extended to work with multiple ground vehicles. Intuitively, more agents lead to improved efficiency and robustness in completing a sweeping task [39] or a coverage task. Efficiency is improved because more agents can complete the task in less time, and the system is more robust in that the task may yet be completed in the event of a single failure. However, it was quickly shown that "adding covering agents does not necessarily improve coverage time" [16]. Since the goal is to create a Hamiltonian cycle, let the path be represented a simple circle. Examine the result of two agents traveling around the circle and initially spaced apart by one empty cell, as shown in Figure 12. If both agents travel the same direction, then the worst-case coverage time does not appreciably change compared to the single robot case.

Casbeer et al. applied this same concept to the task of forest fire monitoring in [11]. Frequent status updates about a forest fire are crucial for effective fire-fighting techniques and for the safety of those involved in the task. Casbeer presents a multi-UAV approach to monitor a fire's perimeter and retrieve updates on the fire's progress. It is assumed that the fire's area is much larger than the communication range of the UAVs, so the base station will only be periodically updated with the new information.

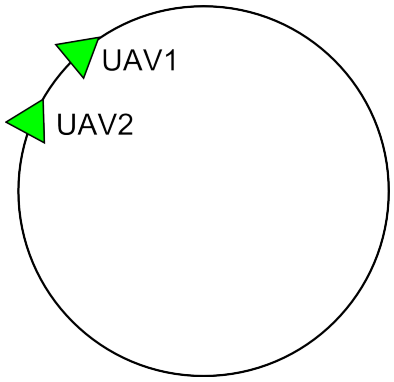

Figure 12: Two UAVs that are slightly spaced apart and travel the same direction have a worst-case coverage time nearly equal to the single vehicle case.

For this task it is important to minimize the worst-case delay of information flow from a checkpoint to the base station. The UAVs fly in opposite directions around the fire's perimeter until a rendezvous with another UAV is achieved. Then, the UAVs exchange data locally and turn around to fly back in the opposite direction, as depicted in Figure 13. When a UAV is within communication range of the base

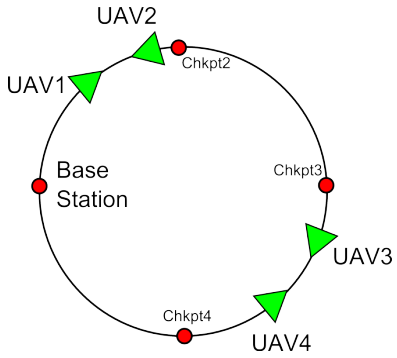

Figure 13: The UAVs exchange data and change directions at each rendezvous, thus achieving a lower worst-case time of getting information about a certain checkpoint back to the base station.

station it transmits all of the collected data to the base. This technique reduces the maximum time delay for information from the fire's perimeter. However, there is a

real amount of time required for a UAV to change directions in flight which is not modeled in Casbeer's system. As the number of UAVs is increased the portion of the time spent turning around after a rendezvous becomes significant. In this document it is assumed that the UAVs remain within communication range of a central base station and therefore the simpler method of all the UAVs flying the same direction may be used without incurring a time penalty. Additionally, the system described in this document creates a complete loop, so vehicles going the same direction helps with deconfliction.

Hazon et al. extended the STC algorithm to incorporate multiple agents in the coverage task [28]. Hazon et al. analytically show that the multiple vehicle spanning tree-based coverage algorithms (MSTC) are robust in that as long as at least one single vehicle is able to move, the coverage task will be completed. However, this relies on the assumption that any failed vehicle will not block the path of the functional vehicles. A failed UAV tends to fall to the ground so the non-blocking assumption remains valid. It is also shown that worst-case coverage times for multi-robots without backtracking are nearly identical to those of a single robot [28]. Hazon et al. go on to show that allowing backtracking improves the overall performance of multi-vehicle planning when compared to not allowing backtracking by any vehicle. In fact, if backtracking is not allowed, the coverage has a worst-case time nearly identical to that of a single robot. However, when backtracking is allowed the work time for coverage is guaranteed to be at least half that of the single robot case. Further, Hazon's algorithm takes into account the starting positions of the agents and shows that for a given spanning tree-based algorithm, this new spanning tree construction method will match or beat the performance of others.

Even et al. present a rooted tree cover algorithm such that the maximum length of k tours over a graph is minimized [20]. Their motivation was for the nurse station location problem in which k nurses are assigned a set of patients to visit during their rounds. The objective is to assign patients to nurses and locate the nurse stations such that the latest completion time is minimized [20]. Of course, this problem is closely related to the multi-vehicle coverage problem as well. The multi-robot forest coverage (MFC) algorithm is based on this work and additionally "tends to return the robots close to their initial cells" [49]. MFC performs closer to optimal for most cases compared to MSTC, but it does not allow backtracking, which reduces its performance in specific environments such as a hallway. Arkin et al. develop the min-max vehicle routing even further in [8]. They present algorithms for a min-max tree cover and min-max star cover, which are useful for the multi-UAV surveillance task.

Agarwal formulates the aerial surveillance task well: Given a planar workpiece R, the objective of region coverage is to find an ordered list of waypoints and the geometry of paths between consecutive waypoints along which the centroid of a sensor footprint can be moved to efficiently trace a minimal superset of R [1]. He points out that while low altitude flight by the UAV helps to avoid occlusions potentially caused by cloud cover, it also restricts the field of view (FOV) since the FOV is directly proportional to the square of the sensor height above the desired surveillance area. He considers n UAVs for coverage over a contiguous holed rectilinear workspace. The workspace is assumed to be rectilinear and the flight controls are simplified by allowing only 90 or 180 degree turns. These assumptions are useful to make the search problem more tractable.

Agarwal assumes that each UAV is equipped with the same type of sensor (homogeneous), but allows for differences in the aircraft frame and dynamics (i.e., the UAVs could vary in speed capabilities). Additionally, it is assumed that the mission altitude is sufficient such that the UAVs fly at a constant altitude, the ground variations do not cause occlusions, and no ground obstacles pose an immediate danger to the aircraft. One drawback is that this method does not include the possibility that the aircraft may not be able to turn in the desired radius necessary to follow the generated waypoints. Agarwal mentions a "lowered potential for collision" between UAVs but does not rigorously address this proposition. In Agarwal's work, the areas of the parts assigned to the UAVs are in the ratio of the rates at which the UAVs can perform coverage [1].

Several authors point out that the starting positions of the vehicles and the particular minimal spanning tree both have significant effects on the final coverage time. In fact, it seems there exist several places for improvement: initial placement of the UAVs, construction of the initial spanning tree, and route planning based on a given spanning tree.

A consistent problem with much of this research is that it assumes rigid grid-based movement of the vehicles. This assumption may be reasonable for ground-based vehicles but vehicle dynamics and camera geometries must be accounted for with unmanned aerial vehicles. Additionally, it is generally assumed that the vehicles are homogeneous. When planning for heterogeneous aerial vehicles, the velocity capabilities and turning rates of the individual platforms must be accounted for. The technique for coverage presented in [27] models the instantaneous coverage region of a robot as a circle. Then the region of interest is filled in with the minimum number

of circles required for complete coverage and finally, a path is generated to connect all of the circles. This approach is not well suite to aerial vehicle coverage because of the dynamics of a side-mounted camera image when the UAV turns.

It is pointed out in [4] that much of the research devoted to multi-UAV planning does not address the coverage problem. Instead, challenges such as formation flight are more frequently addressed [9]. Another issue that is infrequently visited is the projection of a rigidly fixed camera when the UAV is turning. The area viewed by the camera obviously changes as the UAV proceeds through a turn, and this change in the viewed area must be accounted for in the path planning stage [4].

In [40], Maza and Ollero present an algorithm that plans for a small team of heterogeneous UAVs in near real time. The system sub-divides the region based on the capabilities of each UAV, taking into account features such as fuel consumption, maximum flight time, flight speed, sensitivity to wind, and sensing width. For N UAVs, the ROI is partitioned into N polygons using $N-1$ lines. The system uses a lawn-mower type path within each sub-division of the original ROI. They show that it is only necessary to check orientations of the pattern that are perpendicular to the edges of the perimeter of the polygon to minimize the final number of turns. They propose that the system could quickly adapt if one UAV were to fail based on the efficiency of the algorithm. However, the simple replanning method leads to overlap once the replanning occurs. One advantage of their proposed method is that the turning cost of the UAVs is reduced since each UAV determines its orientation independently of the rest of the team. The simple scenario depicted only accounts for convex polygonal ROIs without obstacles. The spanning tree method handles obstacles and many arbitrary shapes by utilizing techniques developed for graph theory.

Completeness of coverage is desirable, but another aspect of multiple vehicles is that the robustness is also increased. In [29], Hazon et al. present a system for multi-robot on-line coverage. The problem is that many of the multi-robot planning algorithms do not achieve complete coverage if a single robot fails unless the task is replanned with the remaining functional robots. Instead, their technique uses an on-line spanning tree coverage algorithm to create a set of paths such that the union of the paths is "guaranteed to be complete, non-redundant, and robust" [29]. The idea is that the coverage path is cyclic so even if one robot fails, the entire region will still be covered by the remaining members of the team. Again, the algorithm is not directly applicable because the robots are assumed to make only 90 degree turns. However the concept of a single cyclic path for the UAV team is especially useful since communications may be even more constrained in aerial vehicles.

Successful cleaning robots have been demonstrated using heuristics and template patterns to achieve complete coverage [33]. The advantage of using templates is that the required computation is reduced. There are several methods to partition a region of interest for multiple vehicle coverage. One of these is a dynamic polygonal partitioning scheme that is presented in [36].

2.3 Heterogeneous Vehicle Teams

The previously described work focused on homogeneous teams, but an interesting challenge is to leverage the strengths of a heterogeneous team of UAVs.

UAVs have desirable features such as high speed of coverage and a wide-area view of the region, while certain tasks such as precise localization may be better suited to ground-based vehicles. One way to combine the best features of both ground- and

aerial-based vehicles is to examine a heterogeneous team. The aerial vehicles may quickly overfly a region to find the general location of an object of interest and then pass the location of that object to a team member on the ground. The ground vehicle then performs a search space over a much smaller area dictated by the uncertainty of the localization of the UAV. The ground vehicle is then able to report back a precise location when the object is found. In [25] and [26], such a framework is described that allows a heterogeneous team of air and ground vehicles to work cooperatively to improve the localization of ground targets. In their test, the air vehicles achieve broad area coverage quickly, but without high resolution on localizing the ground targets. The ground-based vehicles complement those abilities in that they achieve higher resolution in localization of the targets. Their system leverages the abilities of a single UAV to aid a single ground-based vehicle to discover and locate targets on the ground. The research proposed in this document adds to this system by generating paths for multiple aerial vehicles over the initial search space.

CHAPTER III

GENERATING THE GROUND PATH

The multi vehicle coverage planner is similar to work by Choset [15, 13, 14], Gabriely [21, 22], Hazon [28, 29], and Zheng [49]. The research presented here extends this previous work. The primary purpose of the ground path is to provide an ordered list of waypoints which the centroid of the camera's image is to follow. Two techniques are presented to segment the ROI and generate a ground path. The first method makes use of tiles that are parameterized to be adjusted to fit in a given segment. The method is able to incorporate a wide variety of tiles, however a small subset of patterns is the most efficient. The second method makes use of spanning trees to generate the ground path. The tile based method is computationally efficient since many of the parameters are pre-computed. Therefore it is most useful in situations where computation time incurs a higher penalty. The spanning tree area coverage method is more robust since it creates a single tour for the ROI that each of the UAVs follow.

3.1 Parameterized Tiles

A tile refers to a unit shape in the parameterized tile system. The system consists of a set of interlocking tiles whose union covers the entire ROI. It is conceptually similar to the tiles or bricks covering a floor or walkway in a solid interlocking pattern.

The parameterized tile method divides the coverage into two distinct and separate problems (divide and conquer approach). Each tile shape is known to offer a path that

provides complete coverage of its own shape. Therefore, a collection of tiles whose union covers the entire ROI also ensures complete coverage along the resultant path. The primary advantage of the tile-method is that much of the required calculations are pre-computed or are inherent to the shape of the tiles. Therefore, the path calculations can be performed quickly on the fly.

3.1.1 Problem Description

Given a bounded ROI and N UAVs with which to perform coverage of the region, the ROI is divided into N sub-regions with one UAV tasked to each sub-region. The segmentation process takes into account the tile sets that are available to make path planning more readily computed. The goal is also to divide the regions as equally as is conceivable among the set of UAVs.

3.1.2 Tile Set

The *One pass* tile is the simplest of the tiles. A UAV traverses in one direction only across a small region that is only as wide as the footprint of its onboard sensor. That is, the entire tile region is adequately covered by one simple pass.

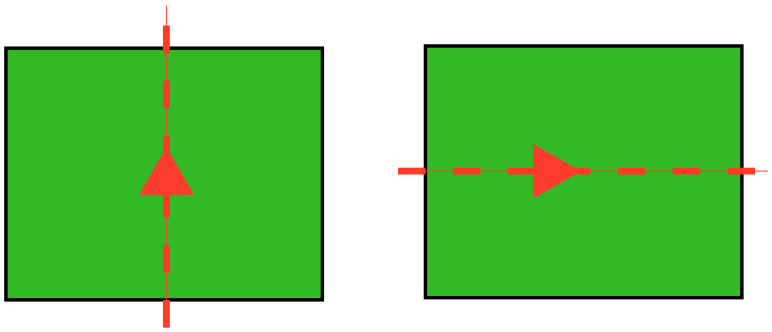

Figure 14: The simplest tile form is just a line through a given region.

The *Two pass* tile is used when the ROI is wider than the attached camera's footprint but still less than three times its width. At three times the width, the region can be covered with a one pass and a perimeter tile. It is a simple form of the *lawnmower* pattern but merits a separate category for reduced complexity in calculating this specific case.

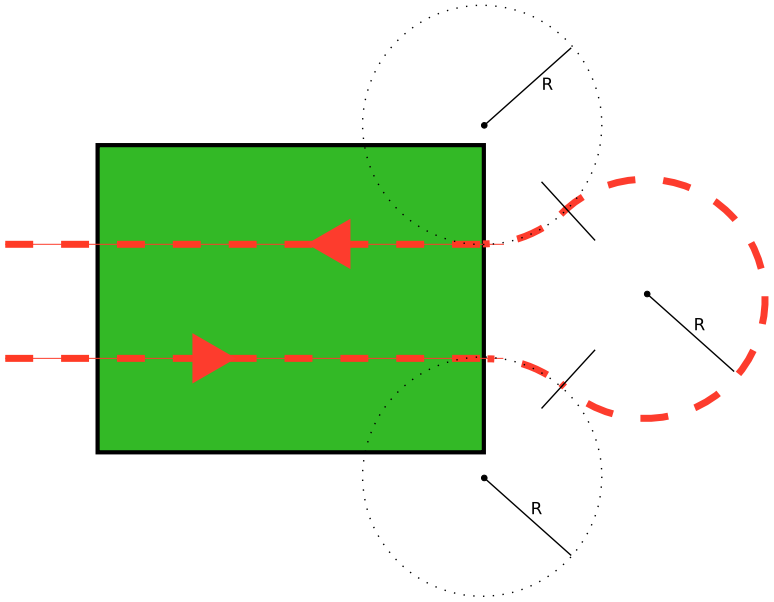

Figure 15: The two pass tile with geometric marking indicating the path construction.

The *Perimeter* tile is a path that follows the inner circumference of the ROI. Side mounted cameras on a fixed wing vehicle are useful to perform a perimeter pattern. Additionally, if it is desired that a target is contained within a known region then the perimeter tile is more heavily weighted to prevent the target from escaping from the region.

The *Lawnmower* tile is the familiar back and forth pattern that one might utilize when mowing a lawn. It is the most frequently used tile while employing the

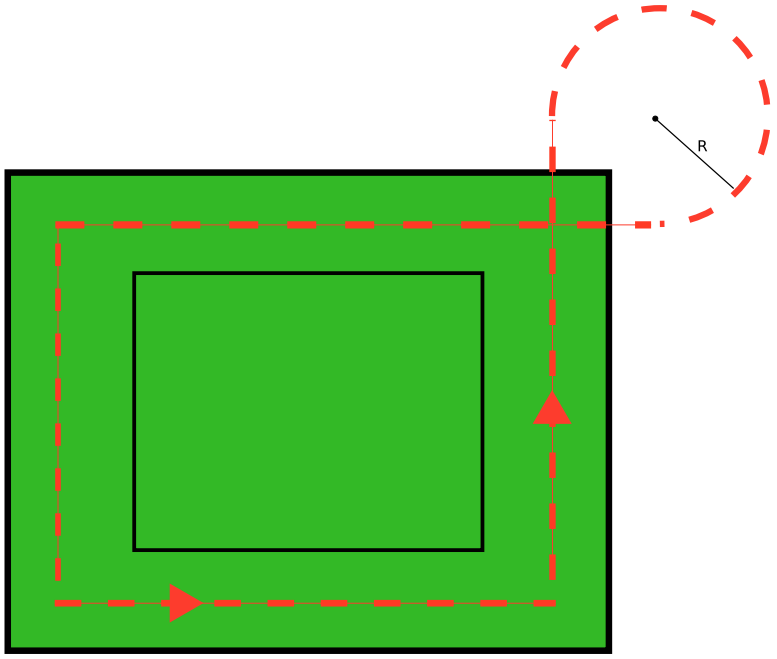

Figure 16: The perimeter tile with two possible turn configurations and geometric markings indicating the path construction.

parameterized tile method.

The *Spiral* pattern is useful for scanning the vicinity of the center of a ROI [46]. Spirals patterns are useful for tasks that involve search from an initial starting point, such as finding victims of water accidents [44]. However, weakness of spiral patterns is in creating a full cycle for the path. A vehicle that begins at the periphery and finishes in the middle must return back to the periphery to begin the spiral again. Coupled with the non-linear dynamics of the camera's relative ground footprint while the vehicle flies a continuous turn, this tile is less practical than the previously described tiles when applied to wide area surveillance.

3.2 Spanning Trees

The minimum spanning tree problem is well known in the area of computer science. It has been used extensively in communication networks and a variety of other computer networks [24]. It is often utilized for routing physical cables to minimize the total cable length, thereby reducing the costs of a project. Solutions to the minimal spanning tree problem have been invented and re-invented numerous times in the last half-century. More recently it has been formalized on graph theory with a few well-defined algorithms. Prim's algorithm [43] and Kruskal's algorithm [38] are the most commonly recognized of the structured algorithms.

Use of a spanning tree for area coverage has also proved useful and computationally efficient. Carmi et al demonstrate in [10] that the minimum spanning tree of a set of points is a constant-factor approximation for the minimum-area spanning tree (MAST) problem.

3.2.1 Problem Description

Given a bounded ROI and some number of UAVs with which to survey the region, generate a single continuous ground tour that completely traverses every cell in the ROI.

3.2.2 Computational Complexity

In general, finding the minimum spanning tree on a graph with V vertices and E edges has been shown to be $O(V^2)$ or $O(Elog(E))$, depending on the method used. The method presented in this document is as fast as comparable implementations. Due to the segmentation of the ROI, every vertex has a maximum of four-connected neighbors (edges). In Chapter 5 this implementation is shown to run in second order

polynomial time.

3.2.3 Spanning Tree Approach to Coverage

The spanning tree area coverage (STAC) algorithm is developed in this document to generate a single, continuous non-overlapping tour over the ROI. As shown in Figure 17 (a), the STAC algorithm works by dividing the ROI into two overlapping grids, one with each cell in the grid being a square of size D, and the other with each cell in the grid being a square of size $2D$, where D represents the camera footprint's size. Figure 17 (b) shows a minimum spanning tree generated over the ROI whose edges connect the centers of the $2D$-sized grid. Finally, Figure 18 shows the path that is generated by connecting the centers of the D-sized grid.

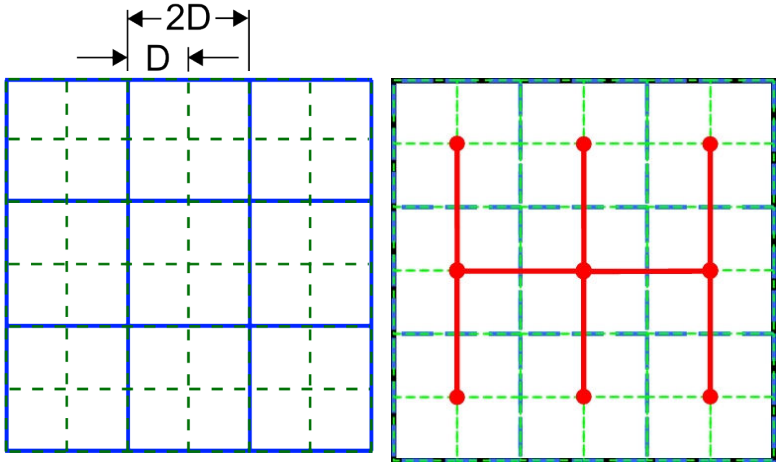

Figure 17: (a) The ROI is divided into a grid with cells of size D and a grid with cells of size 2D. (b) A minimum spanning tree connects the centers of the 2D-sized grid.

The path is generated as a clockwise loop that circumnavigates the spanning tree. For a region described by a single cell, the path would follow in a clockwise loop like the one shown in Figure 19. However, if the transition from one sub-cell to the next is

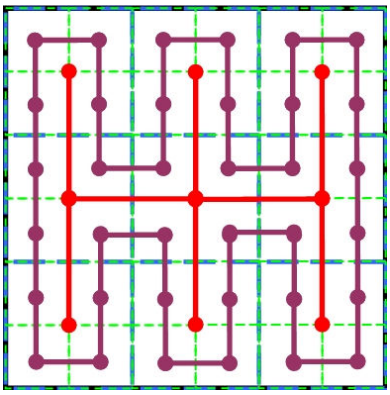

Figure 18: The path connects the centers of the D-sized grid as it follows along one edge of the spanning tree.

blocked by the spanning tree itself, then the path is altered. The same figure displays the modified path through the cell when the spanning tree is taken into account.

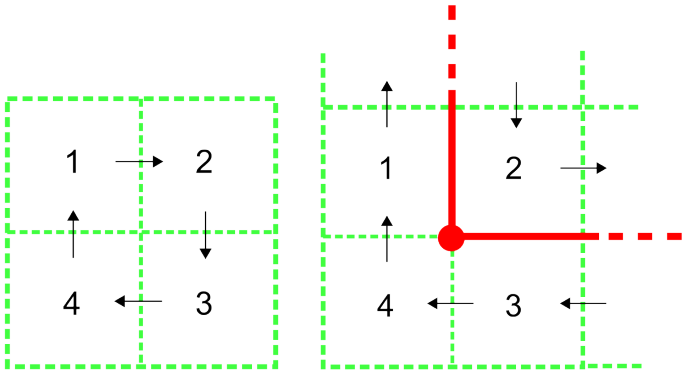

Figure 19: The generated path follows a clockwise pattern around the cells. The right image displays the path through a cell when the spanning tree is taken into account.

The STAC algorithm is closely related to the STC algorithm developed by Gabriely [21]. Several versions of the STC algorithm were developed, including an off-line STC and an on-line STC. The off-line version assumes that the vehicle has *a priori* knowledge of the ROI, while the on-line version allows for lack of knowledge about the ROI

and the spanning tree is constructed in real time as the vehicle traverses the ROI [21]. Both of these algorithms are optimal in that they provide complete coverage without repetitive coverage.

The application of the STC algorithm for multi-vehicle area surveillance is a recent research area. The generalization of the single robot STC algorithm to multi-robot systems was first introduced by Hazon and Kaminka in 2005 [28]. If the time required for coverage by one agent is T, then it seems intuitive that ideally N agents could complete the task in T/N time. Of course, in practice it may not be possible to achieve the ideal improvement. In fact, it has been shown that "adding covering agents does not necessarily improve coverage time" [16]. Therefore, the additional vehicles must be included with a focus on maximizing the utility of each UAV or else there is no guarantee of an improvement in the coverage task.

One specific method to improve performance in the multi-vehicle case, when compared to the single vehicle STC algorithm, is to take into account the starting position of each of the vehicles when constructing the spanning tree. The starting positions of the vehicles have a significant impact on the coverage time of the terrain [2]. If the spanning tree is fixed, but the starting positions of the UAVs are varied, then the time to completion of coverage can vary greatly. The spanning tree that realizes the shortest time to complete coverage is generated by taking into account the UAVs' starting positions.

The minimal spanning tree for a set of vertices is a minimum set of edges that connects all of the vertices without creating any loops. There is not one unique set of edges that is the singular minimum spanning tree for any ROI. Rather, there exists a very large number of combinations of sets of edges that are minimal spanning

trees. The STAC algorithm uses a min-max distance metric to generate one of the spanning trees that realizes a quicker completion time for the coverage task. To create a minimum spanning tree, one must first set up a graph, G. The distance between the vertices of G is $2 * D$, where D is the width of the camera footprint projected onto the coverage area. The spanning tree is then created with an algorithm that is closely related to Kruskal's algorithm [38].

When applied to path planning, the vertices are created by overlaying a grid on the region of interest. Figure 20 shows a grid region with the minimum spanning tree in red. The grid spacing is determined by the projected footprint area of a downward looking surveillance camera. The path followed by a vehicle circumnavigates the spanning tree and is also shown in the figure.

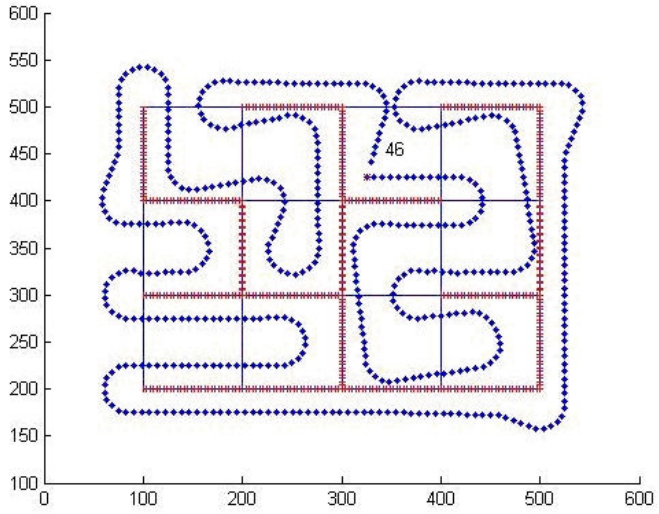

Figure 20: Example simulation of single UAV path planning based on a minimum spanning tree

For the task of aerial surveillance, the path constructed by this technique represents the path that the centroid of the camera footprint will follow. A transformation

based on the camera mounting angle results in waypoints for the vehicle.

3.2.4 Generating Spanning Trees for N Vehicles

Given N UAVs and a ROI, a sub-tree is constructed for each UAV. The union of all the sub-trees includes every vertex in the graph, but the N sub-trees themselves are not connected. Optionally, one can create $N-1$ bridges between the sub-trees in such a way to create a single minimal spanning tree of the entire graph. The problem formulation, solutions, and performance metrics are presented in the following paragraphs.

The process for generating the N trees assumes that the initial position of each vehicle is already set. Another algorithm may be used to optimize the initial placements of the UAVs before generating the set of spanning trees. The initial placements and the particular minimum spanning tree both have a significant impact on the overall performance of the team of UAVs on the coverage task. Therefore, special emphasis is placed on the generation of the spanning tree.

The problem definition can be more formally stated as follows: Let $G = (V, E)$ represent an undirected graph with vertices, V, edges, E, and uniform positive edge weights. The edge weights are positive and equal because they represent the grid of uniform cells on the ROI. Let $T = \{T_1, T_2, ..., T_N\}$ be a set of trees on the graph where a single tree, T_i, is a set of vertices and edges that form a subset of G. Given N UAVs, create a set of N trees such that the union of all the sub-trees incorporates every vertex in the graph, i.e., $V = \cup_{i=1}^{N} V(T_i)$, and such that the length of each tree is equal. Since it is not always possible for the lengths of the trees to be exactly equal, the goal becomes one of minimizing the maximal length of the trees instead. For this

problem the individual sub-trees do not share any vertices or edges.

Let $P \in V$ represent the set of vertices that are nearest to the starting positions of the N UAVs. For every P_i in P a single edge and vertex are added to the spanning tree at each iteration of the algorithm. The objective is to add the vertex and respective edge to tree T_i that maximizes the minimum distance from that vertex to all other sub-trees T_j, where $i \neq j$. The set of vertices to choose from is the set of vertices that are immediately adjacent to the current tree, T_i, provided the vertex is not already a part of another tree, T_j. In an effort to improve the execution of the algorithm, several data structures are used. There is a list of vertices in T_i that can be attachment points for added vertices. Vertices are added to this list when the vertex is added to the tree and removed from this list when the vertex has no more connected edges. This occurs when all of it's connected edges are already incorporated into T_i or some other tree. The Manhattan distance [41] is measured from each neighbor of T_i to each spanning tree T_j. Then, the neighbor that maximizes the minimal Manhattan distance and the corresponding edge are added to the tree T_i. The Manhattan distance is appropriate because of the four-connected nature of the grid. Continue this process until all the vertices in G belong to some tree T, $V(G) = \cup V(T_{i=1}^{N})$. The algorithm is guaranteed to converge provided that all vertices in the ROI are initially reachable.

Figure 21 depicts a rectangular ROI with the starting locations of four UAVs represented by triangles. Figure 22 depicts the state of the sub-trees after several iterations of the algorithm have run. Finally, a set of balanced sub-trees is achieved. In this case, there are 15 vertices and 4 UAVs so the sub-trees are necessarily not exactly equal in length.

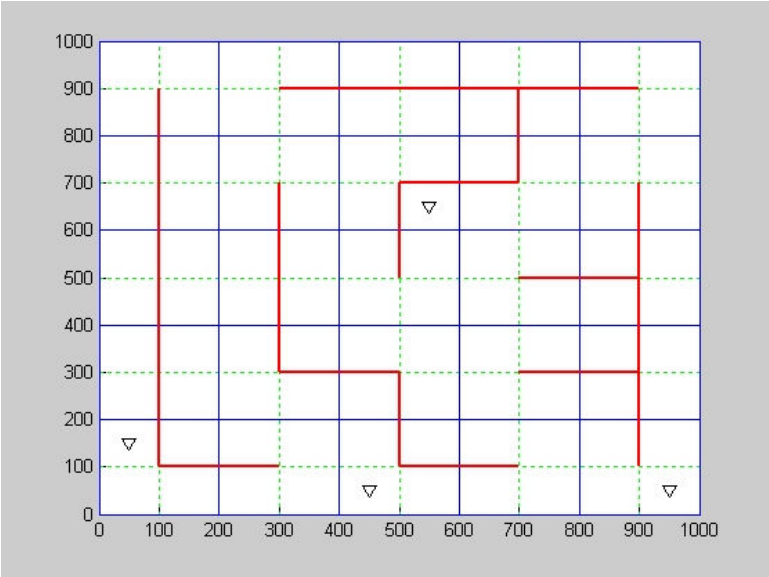

Figure 21: Example of balanced sub-tree generation with four UAVs and their respective starting locations

After the algorithm is run to completion, it is possible to utilize this set of sub-trees to survey the entire ROI. If no further modifications are made, each UAV is assigned to a smaller region of the ROI. These segmented regions are shown in Figure 22. The drawback to using a segmented region is that if a UAV fails, then a gap in the surveillance region needs to be covered by the rest of the team. Two strategies to deal with UAV failure are replanning and creating a single tour.

The replanning strategy to deal with failure is simply to re-apply the STAC algorithm with the remaining $N-1$ UAVs. This option is less than ideal because it can lead to significant portions of the ROI being surveyed multiple times while another region remains unvisited. Therefore one improvement to the replanning method is to track which cells have been visited as they are seen by the UAVs. If a failure occurs, all the previously visited nodes are marked as obstacles and then the replanned paths

are more efficient at covering new territory first rather than already visited territory. While effective, this approach requires more memory and maintained communication with the UAVs.

Creating a single tour that covers the ROI is accomplished by joining all the sub-trees into a single minimal spanning tree. In this case replanning is not necessary if a vehicle fails. Instead, the remaining vehicles continue around the cycle until the lost portion is covered. The time to completion is longer for this method than for the original plan, but the single tour plan completes the first cycle of coverage in equal or less time than the complete replan and reduces the amount of maintained communication that is necessary.

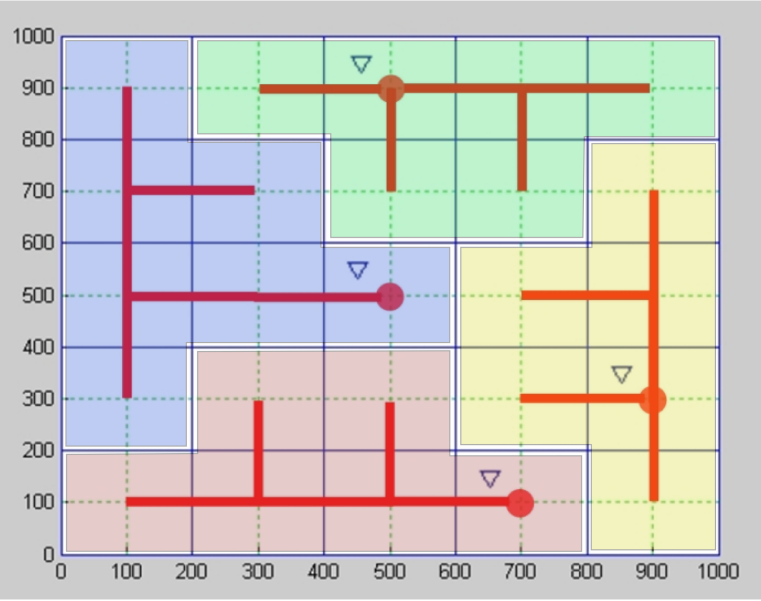

Figure 22: A representation of the ROI segmented into smaller regions, one per UAV.

The choice of joining the sub-trees or not is largely application dependent. Distinct sub-trees may yield better performance for a heterogeneous team of UAVs while

joining the sub-trees is safer if there is an elevated risk of losing a UAV. If the sub-trees should be joined, the problem now becomes how to connect the set of spanning trees, $T \in \{T_1, T_2, ..., T_N\}$, into a single minimal spanning tree while maintaining equal spacing along the cycle formed by the complete spanning tree. A poorly selected bridge between sub-trees could position two UAVs adjacent to one another on the cycle. With one UAV following directly behind the other, the time to complete coverage is approximately the same as if there were only one UAV. The brute force method of examining every possible combination of connections between the trees is too computationally expensive. Therefore, a more efficient method is needed to select the bridge connections.

3.2.5 Joining the Spanning Trees

The work described in the previous section creates a set of balanced sub-trees, with one sub-tree per vehicle. One method to improve the robustness of the coverage algorithm is to join the disparate sub-trees into a single minimal spanning tree. By joining the trees together, a single tour of the entire ROI is created. Since every UAV in the team is following the same tour, complete coverage is maintained even in the event that one UAV fails. Furthermore, no additional communication is necessary in the event of a failure.

As the number of UAVs and vertices increases, so too does the number of potential bridges increase. Examining every possible combination of bridges that join the sub-trees is clearly not an acceptable solution because it is too computationally expensive. Instead, the following paragraphs show that it is only necessary to examine pairings of sub-trees and choose the best bridge for each pair. Given that there are N UAVs,

equation 5 shows the number of comparisons that must be made. Of these, the best $N-1$ are selected to tie all of the sub-trees together.

$$\sum_{i=1}^{N-1} i = \textit{Number of Comparisons} \tag{5}$$

The best bridge for a given pair of sub-trees is the bridge that minimizes the maximum distance between the two UAVs. Let $D_{i,j}$ represent the distance circumnavigating the spanning tree from UAV_i to UAV_j. The time to complete coverage is directly related to the maximal $D_{i,j}$. Therefore, for every pair of UAVs, it is necessary to minimize the maximum distance or, equivalently, minimize the difference between $D_{i,j}$ and $D_{j,i}$. In the simple example shown in Figure 23, there are two possible locations for bridge $B_{1,2}$ labeled A and B. It is clear that if the bridge is located at position A, distance $D_{1,2}$ will be large compared to distance $D_{2,1}$. In fact, UAV_1 and UAV_2 would be nearly adjacent to one another on the resultant tour of the ROI. However, if the bridge at position B is selected then the difference between $D_{1,2}$ and $D_{2,1}$ is much less. In the ideal case, the bridge selection makes $D_{1,2}$ and $D_{2,1}$ equal. By minimizing the maximum $D_{i,j}$, the minimum time to complete coverage is achieved.

Let D_{total} represent the total distance of a path that circumnavigates all of the sub-trees, and let $D(T_i)$ represent the distance of a path that circumnavigates the spanning tree associated with UAV_i. Then it is known that the summation of the distance around each sub-tree is equal to the total distance:

$$\sum_{i=1}^{k} D(T_i) = D_{total} \tag{6}$$

In the ideal case for k UAVs, the distance circumnavigating each sub-tree would be exactly $\frac{D_{total}}{k}$. However, in practice the spanning trees will not be perfectly balanced,

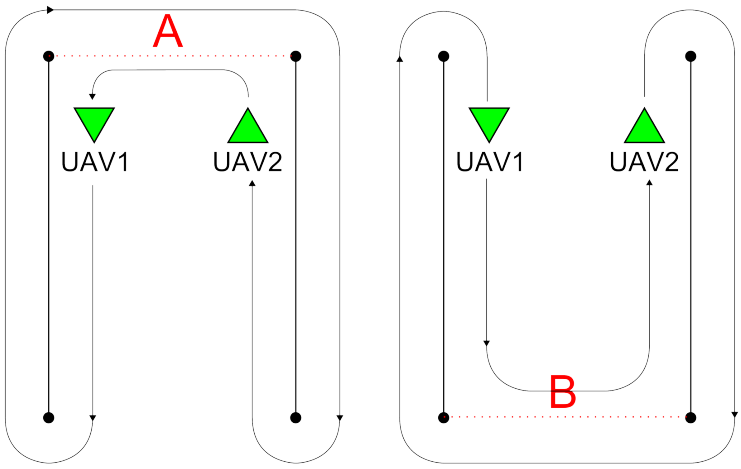

Figure 23: Bridge A makes UAV1 and UAV2 nearly adjacent to one another while Bridge B distributes the UAVs more equally along the resultant path.

but instead will have an error α_i associated with UAV_i. Equation 7 shows the distance of a single sub-tree including the error term.

$$D(T_i) = \frac{D_{total}}{k} + \alpha_i \tag{7}$$

Since the summation of the individual distances around the sub-trees is equal to D_{total}, then all of the error terms must sum to zero.

$$\sum_{i=1}^{k} \alpha_i = 0 \tag{8}$$

As Figure 23 shows, the best bridge connection between two sub-trees might be some error distance away from the ideal case. It can be shown that the general distance, $D_{i,j}$, between any two adjacent UAVs is given by

$$D_{i,j} = \frac{D_{total}}{k} + \left(\frac{\alpha_i + \alpha_j}{2} + \epsilon_{i,j} \right) \tag{9}$$

where $\epsilon_{i,j}$ is the error introduced by a bridge joining sub-trees i and j. It is useful to note that the error terms are only dependent on sub-trees i and j, even if the resulting path runs adjacent to an intermediate sub-tree. Since the goal is to minimize the maximum $D_{i,j}$, and the tree error terms α_i and α_j are already dictated by the current set of spanning trees, $\epsilon_{i,j}$ must be minimized by selecting the best bridge between the two sub-trees.

Joining the sub-trees can be viewed on a higher level as another minimum spanning tree problem on a weighted graph. As depicted in Figure 24, multiple edges indicate that more than one potential bridge exists between those vertices. Let every sub-tree

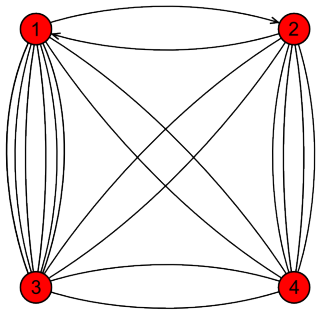

Figure 24: Each sub-tree is reduced to a single vertex and all potential bridges are shown as edges in the graph.

be reduced to a single vertex, and let every potential bridges between the sub-trees be represented by edges connecting the vertices. The edge weights of this graph are represented by the error term associated with that bridge, $\epsilon_{i,j}$. Figure 25 represents the reduced graph with only the minimum edge connections visible.

In fact, joining the sub-trees can be formulated as Kruskal's algorithm [38], which then achieves the minimal spanning tree over the reduced set of nodes. Then, expand the nodes and the result is a single minimal spanning tree with the UAVs spaced as

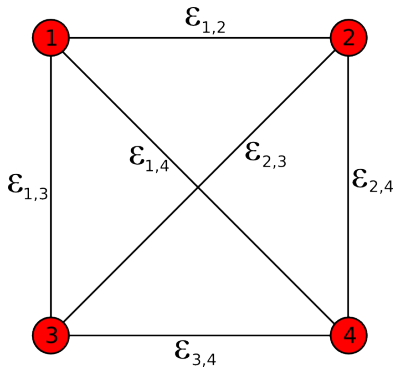

Figure 25: Each sub-tree is collapsed to a single node with edge weights based on $\epsilon_{i,J}$.

equally apart as possible for the given set of sub-trees and starting positions. There is no set of bridges that results in a lower maximum distance between UAVs for the given starting locations and sub-trees. Therefore, the set of bridges selected by the technique described in this chapter provides the minimum time to complete coverage.

CHAPTER IV

WAYPOINT TRANSFORMATION

The previous chapter described methods for generating a continuous ground path targeted at area coverage for an autonomous vehicle. This ground path provides complete coverage for an unmanned ground vehicle (UGV) by visiting every cell in the ROI. The path is not directly usable by UAVs, but can be used as a basis from which to generate an aerial path. The module to transform ground waypoints to aerial waypoints is described in this chapter. It is developed in a generic way so that it is not directly tied to any particular method of ground path generation.

4.1 Justification

The existing ground path is assumed to lie within a 2D plane at ground level. Instead of representing the route followed by the vehicle, let the path represent the route for the centroid of the camera's image over the ROI. That is, the vehicle should fly in such a pattern as to keep the center point of the camera's image along the line of the ground path.

The flight dynamics of a fixed wing UAV result in a non-linear coverage pattern while the vehicle is turning. Therefore, it is necessary to transform the ground path into a set of aerial waypoints that guides the UAV. Merely raising the existing path to flight elevation is not adequate. However, maintaining the new route within a 2D plane is appropriate for aerial vehicles provided that the aerial vehicle operates within a limited range of altitudes that do not follow the terrain's geographic features [5].

4.2 Single Waypoint Transformation

The method to find an aerial waypoint from which to view a particular ground location is detailed in this section. Figure 26 offers a visualization of the angle terms as they are used in this section. Beginning with the upper left image and continuing clockwise, the UAV's pitch angle is represented by θ, the roll angle is ϕ, and the yaw or heading angle is ψ.

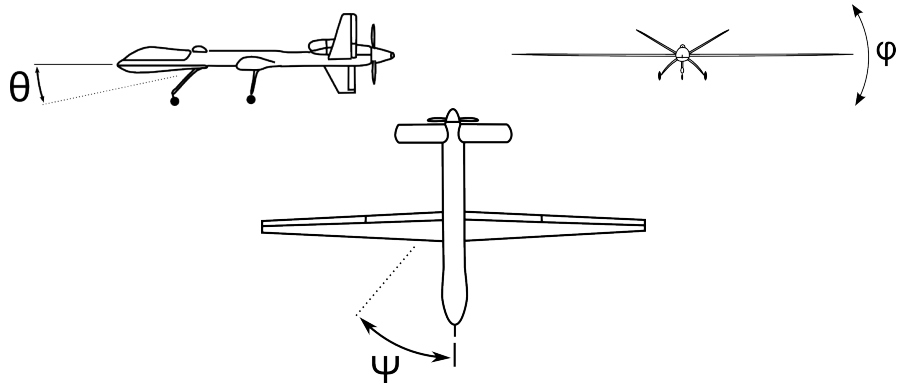

Figure 26: Pitch, Roll, and Yaw angles depicted for a Predator UAV.

The yaw, pitch, and roll rotation matrices for the camera are shown in the following equations:

$$Y_c = \begin{bmatrix} \cos\psi_c & \sin\psi_c & 0 \\ -\sin\psi_c & \cos\psi_c & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

$$P_c = \begin{bmatrix} \cos\theta_c & 0 & -\sin\theta_c \\ 0 & 1 & 0 \\ \sin\theta_c & 0 & \cos\theta_c \end{bmatrix} \tag{11}$$

$$R_c = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi_c & \sin\phi_c \\ 0 & -\sin\phi_c & \cos\phi_c \end{bmatrix} \tag{12}$$

where θ_c is the mounting angle of the camera with respect to the vehicles frame. $\theta_c = -90°$ indicates that the camera is facing left, $\theta_c = 0°$indicates forward facing, and $\theta_c = +90°$ indicates that the camera is right facing relative to the forward direction of the UAV. ψ_c is the camera's yaw angle with $\psi_c = 0°$ pointing directly down and $\psi_c + 90°$ pointing in the direction of travel of the UAV. Finally, the camera's roll angle, ϕ_c, is assumed to be $0°$ for this application. The roll angle is important for interpreting the resultant images and for tasks such as geo-registration. However, the camera's roll angle does not affect waypoint positioning.

For the UAV,

$$Y_{uav} = \begin{bmatrix} \cos\psi_{uav} & \sin\psi_{uav} & 0 \\ -\sin\psi_{uav} & \cos\psi_{uav} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{13}$$

$$P_{uav} = \begin{bmatrix} \cos\theta_{uav} & 0 & -\sin\theta_{uav} \\ 0 & 1 & 0 \\ \sin\theta_{uav} & 0 & \cos\theta_{uav} \end{bmatrix} \tag{14}$$

$$R_{uav} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi_{uav} & \sin\phi_{uav} \\ 0 & -\sin\phi_{uav} & \cos\phi_{uav} \end{bmatrix} \tag{15}$$

where ψ_{uav} is the yaw or heading angle, θ_{uav} is the pitch angle, and ϕ_{uav} is the roll angle.

To find the transformed waypoint, let V represent a unit vector pointing straight out of the camera's lens. Vector V is, necessarily, pointing directly at the center pixel of the camera's image. The location of the camera needs to be translated and rotated such that the center pixel falls on the desired line path on the ground. Let W_j' be an aerial waypoint from which the camera views the single waypoint, W_i. Equation 16 transforms V relative to the vehicle's frame, and then transforms the vehicles frame with respect to the global ground frame, such that V points toward the ground location specified by W_i.

$$V' = [Y_{uav} \times P_{uav} \times R_{uav}] \times [Y_c \times P_c \times R_c] \times V \tag{16}$$

The final step in finding the new waypoint is to scale and translate V as shown in Equation 17. Now W_j' represents the location where the UAV needs to be for the ground point to be in the center of the camera image.

$$W_j' = \frac{V'}{h} - W_i \tag{17}$$

4.3 Whole Path Transformation

Individual waypoints can now be re-positioned, but more planning is required to transform the entire path. The mapping from ground waypoints to aerial waypoints is a one-to-one mapping only in the case of a straight line. For turns, additional waypoints need to be inserted to ensure a smooth resultant path. Therefore, turns in the underlying ground path need to be identified and located.

Let W be the set of ground waypoints representing the current ground path and let W_i represent the i^{th} waypoint in the set. At least three waypoints need to be

examined to determine the type of turn described by the waypoints. Utilizing four waypoints allows further path optimizations to be implemented. The turns in the ground path are identified and classified by considering a local subset of only four waypoints, $(W_i, W_{i+1}, W_{i+2}, W_{i+3})$.

4.3.1 Types of Paths

The set of four waypoints is divided into two overlapping sets of three waypoints. Each set of three waypoints is labeled as one of three classifications: straight line, simple left, or simple right. Figure 27 shows an arbitrary set of three waypoints in a 2D plane. Arrows in the figure depict direction of travel and relevant angles are also labeled.

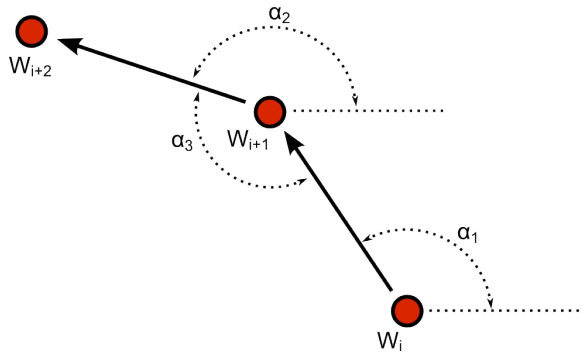

Figure 27: Three arbitrary waypoints in 2D space.

For any three arbitrary waypoints, the following equations find the angles α_1, α_2, and α_3 (where X and Y indicate components of the ground coordinate grid).

$$\alpha_1 = \tan^{-1}\left(\frac{Y(W_{i+1}) - Y(W_i)}{X(W_{i+1}) - X(W_i)}\right) \tag{18}$$

$$\alpha_2 = \tan^{-1}\left(\frac{Y(W_{i+2}) - Y(W_{i+1})}{X(W_{i+2}) - X(W_{i+1})}\right) \tag{19}$$

$$\alpha_3 = \pi + \alpha_1 - \alpha_2 \tag{20}$$

Assuming that all the angles are wrapped to the interval $0 \leq \alpha \leq 2\pi$, then α_3 is used to classify the type of turn indicated by this set of waypoints. A left turn is indicated by $0 < \alpha_3 < \pi$ and $\pi < \alpha_3 < 2\pi$ indicates a right turn in the ground path. If $\alpha_3 = \pi$ then the three waypoints are on a line.

If the distance between W_{i+1} and W_{i+2} is greater than the turning radius, R, then this is classified as a simple turn with α_3 indicating the direction of the turn. However, if the distance is less than R, the next set of waypoints, $(W_{i+1}, W_{i+2}, W_{i+3})$, is also considered. If α_3 for the second set of waypoints falls within the same range as the first set then it is classified as a compound turn. The following sections describe how the waypoints are modified for each type of turn.

4.3.2 Straight Lines and Simple Turns

The new aerial waypoints must be calculated based on their previous classification. The most simple classification is that of a straight line containing several waypoints. The transformation to aerial waypoints consists of repeated applications of the single waypoint transformation with the UAV's heading set equal to α_1.

The distance, D_{xy} in the x-y plane between W_i and W_j' is important in determining the number of additional waypoints necessary to accommodate turns. As noted in Equations 16 and 17, this linear distance is dependent on the flight altitude of the UAV as well as on the specific geometries of the associated camera.

The simpler case occurs when the turning radius of the UAV is less than the linear distance, as depicted in Figure 28. In this case, W_i is transformed with $\psi_{uav} = \alpha_1$ and W_{i+2} is transformed with $\psi_{uav} = \alpha_2$. A temporary waypoint, W_0, is useful in constructing the remaining points for the simple turn. If one were to imagine two

lines, one passing through W'_j with angle α_1 and the other passing through W'_{j+3} with angle α_2, then W_0 is the point where these two lines intersect. W'_{j+1} and W'_{j+2} represent the waypoints where a circle of radius R is tangent to each of the lines.

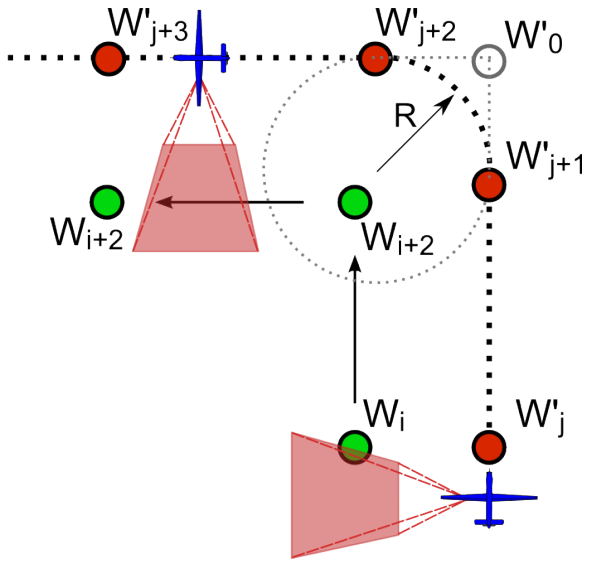

Figure 28: If the turn radius of the UAV, R, is small enough a simple left turn is applicable.

Equations 21 and 22 are used to calculate the locations of the intermediate two waypoints.

$$\begin{aligned} X(W'_{j+1}) &= X(W_0) + T * \frac{R * \cos(\alpha_1)}{\tan(\alpha_3/2)} \\ Y(W'_{j+1}) &= Y(W_0) + T * \frac{R * \sin(\alpha_1)}{\tan(\alpha_3/2)} \end{aligned} \tag{21}$$

$$\begin{aligned} X(W'_{j+2}) &= X(W_0) - T * \frac{R * \cos(\alpha_2)}{\tan(\alpha_3/2)} \\ Y(W'_{j+2}) &= Y(W_0) - T * \frac{R * \sin(\alpha_2)}{\tan(\alpha_3/2)} \end{aligned} \tag{22}$$

T is -1 for a left turn and $+1$ for a right turn. The center for the circle that describes

the turn radius of the UAV follows.

$$X_{cc} = X(W'_{i+1}) + T * R * \sin(\alpha_2)$$
$$Y_{cc} = Y(W'_{i+1}) - T * R * \cos(\alpha_2) \tag{23}$$

Figure 29 shows the case when the UAV's turning radius is larger than D_{xy}.

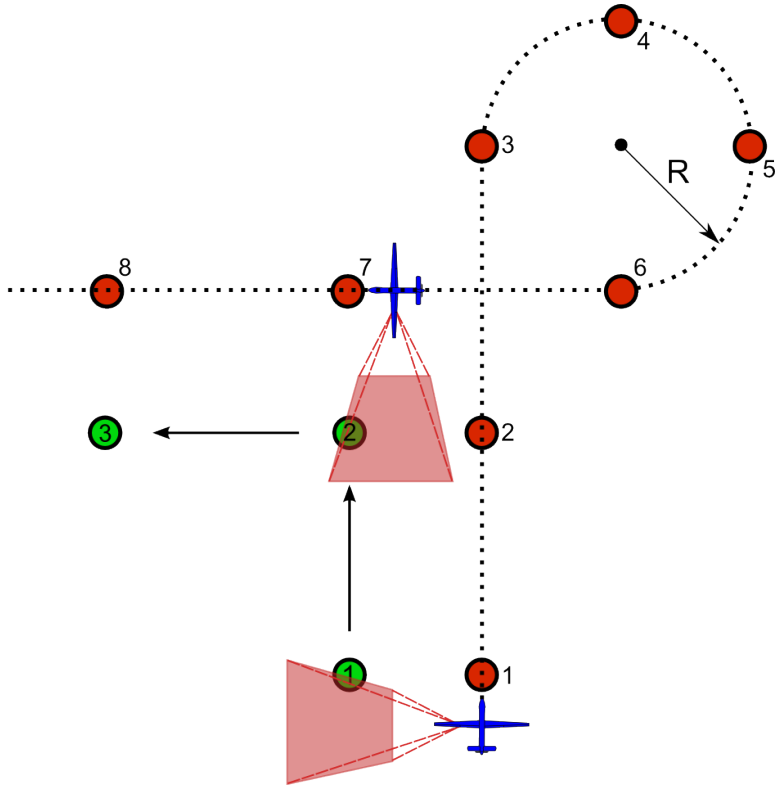

Figure 29: If the turn radius of the UAV, R, is large then additional waypoints are necessary to ensure that the camera's image follows the ground path accurately.

In this case the UAV must make a wider turn in the opposite direction to ensure that the camera's footprint lines up correctly with the ground path. This is also necessary, for example, when a UAV outfitted with a rightward facing camera makes a left turn. Obviously, these turning maneuvers add cost to the overall completion time.

Therefore, minimizing the number of turns increases performance, but completeness of coverage should not be compromised.

The previously defined equations are again employed to locate the new waypoints in this example, with the only difference being that the sign of T is inverted.

4.3.3 Compound Turns

What happens if there are two turns in close proximity? If the turns are left then right, or right then left they are treated just like individual 90 degree turns. On the other hand, if there are two right-hand turns or two left-hand turns adjacent to one another, then a special u-turn case is needed.

For the u-turn case, the UAV can fly the circumference of a circle if it is within the flight envelope of that UAV. If not, it must fly along the circumference of a circle that is positioned outside the immediate ROI, as shown in Figure 30. This ensures

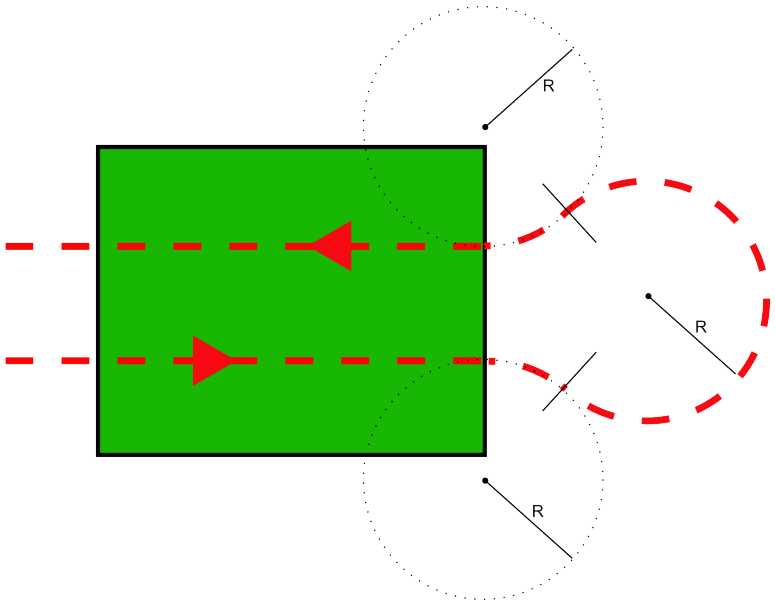

Figure 30: A compound turn has a shorter path than if the two turns were connected by a cloverleaf.

coming back into line with ground path after temporarily leaving it. Figure 31 shows the simulated trajectory of a UAV using all of the previously described methods.

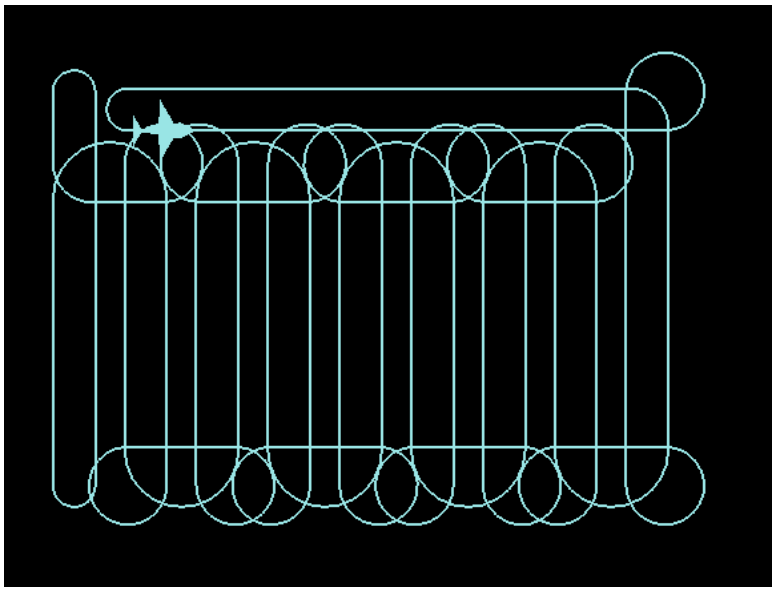

Figure 31: Simulation of the path of a UAV showing straight lines, simple turns, and compound turns.

CHAPTER V

COMPLEXITY ANALYSIS

The previous chapters describe a useful algorithm to generate a ground path for several vehicles and to transform that path into a set of suitable aerial waypoints for UAVs. For this algorithm to be practical, it needs to be efficiently scalable in terms of the size of the region to be examined as well as in the number of UAVs assigned to the surveillance task. The algorithm must scale for an increasing number of vertices as the size of the ROI increases as well as for an increasing number of UAVs for improvements in performance.

Computational complexity theory in computer science describes the scalability of an algorithm. The results obtained from a careful analysis are relevant to how well a given algorithm scales in a practical application. The target application for the algorithms described in this document is surveillance by small-sized UAVs. Due to the limited amount of on-board computational power typically found on small UAVs, it is especially important to keep the algorithm's complexity low, or to utilize heuristics to ensure that the execution time remains reasonable.

5.1 Assumptions and Reasoning

The following paragraphs describe the computational complexity analysis of the algorithm presented in this document. The basic assumptions are that a generic one-processor random-access machine (RAM) model is used. No concurrent operations are allowed in this analysis. Let $G(e, v)$ represent a graph, G, with vertices, v, that

are connected by edges, e. Let E represent the total number of edges, e, and let V represent the total number of vertices, v. The primary inputs for this analysis are the total number of UAVs, K, and the total number of vertices, V, in the graph used to generate the spanning tree. Since the size of the ROI and the camera's geometry both are used to determine the total number of vertices, they are indirectly inputs as well.

The upper bound of the running time of the algorithm is the most pertinent metric of this analysis, thus it is expressed in the common O-notation. For a given function $f(n)$, the O-notation of that function, $O\left(f(n)\right) = g(n)$, indicates that there exists positive constants c and n_0 such that $0 \leq g(n) \leq c \cdot f(n)$ for all $n \geq n_0$. For example, Figure 32 displays this relationship graphically for the function $g(n)$ when $f(n) = n^2$.

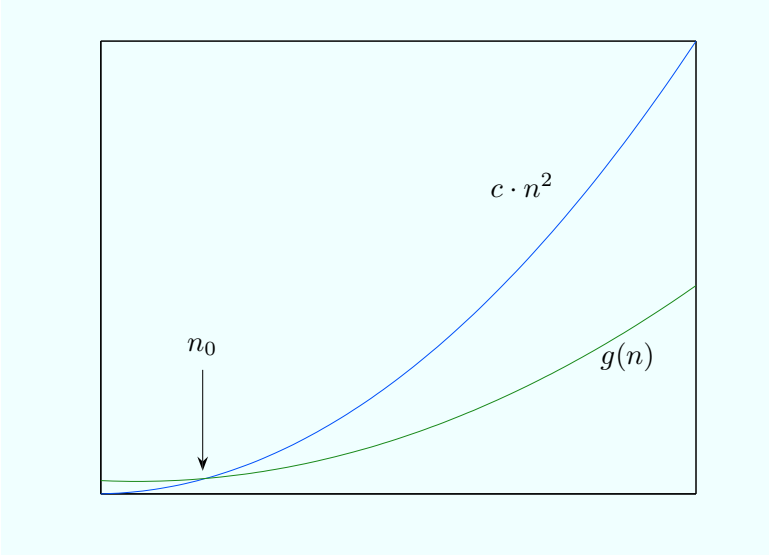

Figure 32: An example of O-notation: Let $g(n) = O(n^2)$, then $g(n)$ has an upper bound defined by $c \cdot f(n)$ for all $n \geq n_0$

5.2 Parameterized Tile Approach

A benefit of the parameterized tile approach is that much of the computation is inherent in the tiles themselves.

The parameterized tile approach uses heuristics to divide the surveillance region into cells and then assign one cell per UAV. This approach does not guarantee an optimal path solution, but it requires little computational overhead compared to other methods.

5.2.1 Segmentation and Tile Selection

The shape of the available tiles plays a role in the segmentation process. The heuristics that affect the tile selection are mainly related to the camera's geometry and mounting angles. The first tile is always a perimeter tile if there are multiple UAVs. The ratio of the perimeter distance to the interior area determines how many perimeter rings are selected before dividing the remaining interior into other tile shapes. Since one tile is assigned to each UAV, the segmentation process is linear in the number of UAVs, $O(K)$.

With the runtime for the segmentation process known, the run-time for each tile should be analyzed. The perimeter tile runs in constant time, $O(1)$, for ROIs that have a regular shape. If the ROI has jagged edges or an irregular shape, in the maximum-time case it could be linear in the number of vertices in the graph, $O(V)$. The lawnmower tile also generates a regular pattern. Its run-time in the maximum-case would cover the entire ROI. Therefore, its run-time is also linear in the number of vertices, $O(V)$. Using a similar approach, the spiral tile also could run in linear time, $O(V)$, if required to span the entire ROI. Since one tile is assigned to each UAV,

the maximum overall run time for parameterized tiles is $O(VK)$.

5.3 Spanning Tree Approach

The spanning tree approach utilizes multiple steps that will be analyzed independently. Then, the results are combined for the completed analysis. First, the region is segmented into uniformly sized cells. Then, a directed spanning tree is generated for each UAV over the resultant graph. From these disjoint spanning trees, ground waypoints that circumnavigate the trees are collected to achieve an effective ground path. After this step, the disjoint trees are connected via bridges such that the UAVs are equally spaced out along the resultant path. With the new bridge connections in place, the order of the ground waypoints is once again collected. Finally, aerial waypoints are generated from the ground path. The following sections analyze each major portion of the algorithm.

5.3.1 Segmentation of the Region of Interest

The ROI is segmented into regular rectangular cells whose sizes are determined by the footprint of the UAV's camera projected onto the ground. Each cell is represented by a single vertex in a graph, G. During the segmentation process, the vertex for every cell is visited one time to establish the initial edge connections in the graph. Assuming a rectangular arrangement, every vertex is connected to its neighbor to the right and its downward neighbor, as shown by Algorithm 5.1. This generates a graph with at most four connections per vertex. This is also true for ROIs with irregular shapes, that no vertex has more than four edges.

For a regular region, out of bounds vertices are those that lie past the last row or column. For an irregular ROI, out of bounds is any portion not contained entirely

Algorithm 5.1 Make Edge Connections

$r \leftarrow$ *number of rows*
$c \leftarrow$ *number of columns*
for $n = 1$ to $length(vertices)$ **do**
  add edge from $vertex[n]$ to $vertex[n + r]$ unless $vertex[n + r]$ is out of bounds
  add edge from $vertex[n]$ to $vertex[n + 1]$ unless $vertex[n + 1]$ is out of bounds
end for

within the desired ROI. Vertices at the perimeter of the region or adjacent to an obstacle will have fewer than four-connected edges. Additional labels are added to the vertices and edges which aid in computation later in the algorithm.

Since each vertex, v, is visited exactly one time to create the edge connections, the computational complexity is linear in V, $O(V)$. Adding labels is also a linear function of vertices, V, and edges E. The edge connections and labeling functions are combined to run in $O(V + E)$.

5.3.2 Building the Directed Spanning Trees

The directed spanning tree is built upon two sub-functions, $GatherSeedVertices()$ and $AddVertexToTree()$. These two sub-functions must be analyzed independently to understand their contribution to the complexity of the more general algorithm. The purpose of the first sub-function, $GatherSeedVertices()$ is to generate a list of vertices that are one-connected to a UAV's spanning tree but are not connected to any other UAV's spanning tree. The pseudo-code for $GatherSeedVertices()$ is shown in Algorithm 5.2.

At most there can be four vertices connected to any vertex. Therefore, the running time for this portion is linearly bounded by the number of vertices in the UAV's spanning tree. On the first iteration, there is only one vertex in the tree. One

Algorithm 5.2 Gather Seed Vertices

```
for n = 1 to length(Tree) do
  if out_degree(vertex[n]) > 0) then
    for each vertex, v connected to vertex[n] do
      if v ∉ anyTree then
        seed_vertices ← v
      else
        remove edge, e, from vertex[n]
      end if
    end for
  end if
end for
```

additional vertex is added to the tree on each successive iteration. Therefore, the number of vertices is dynamically variable through the running of the algorithm. On average, the maximum number of vertices is $\frac{V}{K}$, where V is the total number of vertices and K is the number of UAVs. Therefore, a conservative estimate is that it has a running time of $O(\frac{V}{K})$, but in the maximum case the running time could be $O(V)$. Upon completion, Algorithm 5.2 produces a list of vertices that are one-connected to the UAV's tree. The next vertex added to the UAV's spanning tree must come from this list of choices.

Algorithm 5.3 Add vertex to Tree

```
if length(seed_vertices) > 0 then
  for n = 1 to length(seed_vertices) do
    find min distance from vertex to every other UAV's tree
  end for
  for n = 1 to length(seed_vertices) do
    find max of the min distances
  end for
end if
Tree ← MinMaxVertex
remove edges to this vertex
```

The next sub-function is listed in Algorithm 5.3. The purpose of this portion is to find the best vertex to add to the UAV's tree from the list of candidate vertices produced by $GatherSeedVertices()$. The minimum distance from this vertex to every other UAV's tree is calculated. From these distances, the maximum value is chosen. Therefore, at least $K * length\{seed\_vertices\}$ steps are necessary. The number of seed vertices is at most four times the number of vertices in the tree. This is expected to run in less than $O(V - \frac{V}{K})$ but in the maximum case could be $O(VK - V)$. The higher level function built from these two sub-functions is displayed as Algorithm 5.4.

Algorithm 5.4 Build Directed Spanning Tree

```
n ← 1
while nAssigned < (nVertices − nUAVs) do
  seed_vertices ← gather potential vertices for UAV[n]
  result ← add vertex from seed_vertices to tree[n]
  if result > 0 then
    nAssigned ← nAssigned + 1
  end if
  n ← n + 1
  if n ≥ nUAVs then
    n ← 1
  end if
end while
```

This algorithm is not complete until every vertex in the ROI is assigned to the spanning tree of one of the UAVs. At which time, the number of vertices assigned to every UAV is approximately equal. Therefore, at a minimum it must run $V - K$ times since each UAV begins with a single vertex in its tree. For each iteration of the loop, it attempts to add a vertex to a UAV's spanning tree, cycling through the UAV's each iteration. In the maximum-time case, all but one UAV could be boxed in such that the algorithm must loop through K times for each additional vertex

Table 1: Analysis of the Directed Spanning Tree algorithm and its sub-functions

Function Name	Nominal-Time	Maximum-Time
$GatherSeedVertices()$	$\frac{V}{K}$	V
$AddVertexToTree()$	$V - \frac{V}{K}$	$V \cdot K - V$
$BuildDirectedSpanningTree()$	$V + K$	$V \cdot K$
Combined	$V^2 + K \cdot V$	$V^2 \cdot K^2$

to be added, which results in $V * K$ iterations. However, precautions are taken in the positioning of the UAVs to prevent such a situation. Therefore, in practice, the expected number of iterations is $(V + K)$.

Table 1 summarizes the analytical results for nominal and maximum-time scenarios. The number of vertices, V, dramatically outpaces the number of UAVs, K. Therefore, it is clear that the runtime is bounded by a second order polynomial function for all cases.

5.3.3 Generating the Ground Waypoints

The ground path is a set of waypoints that circumnavigates the UAV's spanning tree. The function $GetNextWaypoint()$ shown in Algorithm 5.5 is called repeatedly in the formation of this path. Since at most four edges are connected to any given vertex, this has a constant running time, $O(1)$.

Algorithm 5.5 Get Next Waypoint

```
for i = 1 to out_edges(v) do
   label blocked transitions
end for
if transition ≠ blocked then
   nextwaypoint ← default
else
   nextwaypoint ← alternate
end if
```

The function to generate the ground waypoints, depicted in Algorithm 5.6, loops through all the waypoints in the UAV's tree. Since there are typically $\frac{V}{K}$ vertices, it is expected to run in $O(\frac{V}{K})$. However, it could be as much as $O(V)$ in the maximum case.

Combining the constant run-time of $getNextWaypoint()$ with the nominal or maximum runtime for Algorithm 5.6 results in the same run times.

Algorithm 5.6 Generate Ground Waypoints

```
for i = 1 to K do
  p1 ← startposition
  while p2 ≠ p1 do
    wpt ← getNextWaypoint()
    p2 ← wpt
  end while
end for
```

The previous functions created K sub-trees, one for each UAV, and determined a Hamiltonian path circumnavigating each of the K sub-trees. The goal of the $BuildBridges()$ routine is to interconnect the K sub-trees with $(K-1)$ edges to create a single Hamiltonian path that traverses the entire ROI. The K UAVs should be equidistant along this resultant path. That is, $D_{i,j}$ along the path for all $i \neq j$ should be equal, where $D_{i,j}$ represents the distance from UAV_i to UAV_j along the path. Since perfect equality is not likely, instead the goal is to minimize the error.

5.3.4 Connecting the Bridges

The $BuildBridges()$ function adds a test bridge to the tree and then calculates the error between $D_{1,2}$ and $D_{2,1}$. The number of tests could be up to $4 * V$ iterations of up to V tests. In practice, it is more likely to be $O(\frac{V^2}{K})$ since the nominal number of vertices in each tree is $\frac{V}{K}$.

Algorithm 5.7 Build Bridges

```
for i = 1 to length(UAVs) do
  for j = i + 1 to length(UAVs) do
    distance[k] ← test_bridges()
    bridge[k] ← edge(i, j)
    k ← k + 1
  end for
end for
sort distance from min to max
for i = 1 to length(UAVs) do
  add bridge[i] to tree
end for
```

5.3.5 Generating the Aerial Waypoints

After completion of the ground path, the final step is to transform the waypoint to aerial waypoints. This requires a constant computation for each vertex, v. Therefore, it runs in $O(V)$.

5.3.6 Complexity Conclusions

The parameterized tiles were shown to run in $O(V \cdot K)$ time. Table 2 summarizes the major contributions of complexity to the spanning tree method. The nominal-time case assumes that the final disjoint spanning trees are approximately equal. If this assumption holds true, then the conclusion is that this algorithm runs in second order polynomial time with respect to V and linear time with respect to K. The measured results confirm this expectation and are displayed in Chapter 7. The maximum-time could occur instead if the initial positions of the UAVs are unfavorable. However, the algorithm specifically prevents this situation by removing unfavorable initial starting positions.

Algorithm 5.8 Test Bridges

for $i = 1$ to $length(vertices)$ **do**
  for each out_edge of $vertex[i]$ **do**
    if edge is connected to target UAV's tree **then**
      $test[k] \leftarrow edge$
      $k \leftarrow k + 1$
    end if
  end for
end for
for $i = 1$ to $length(test)$ **do**
  add $test[i]$ to the tree
  $D_{1,2} \leftarrow$ distance from $UAV_1 toUAV_2$ along circuit
  $D_{2,1} \leftarrow$ distance from $UAV_2 toUAV_1$ along circuit
  $diff \leftarrow abs(D_{1,2} - D_{2,1})$
  if $diff < min$ **then**
    $min \leftarrow diff$
  end if
  remove $test[i]$ from the tree
end for
return $min(diff)$

Table 2: Summary of the complexity contributed by various subfunctions of the spanning tree algorithm

Function Name	Nominal-Time	Maximum-Time
$GatherSeedVertices()$	$\frac{V}{K}$	V
$AddVertexToTree()$	$V - \frac{V}{K}$	$V \cdot K - V$
$BuildDirectedSpanningTree()$	$V + K$	$V \cdot K$
Combined	$V^2 + K \cdot V$	$V^2 \cdot K^2$
$GroundWaypoints()$	$\frac{V}{K}$	V^2
$BuildBridges()$	$\frac{V^2}{K}$	V^2
$AerialWaypoints()$	V	V
Total Combined	$O(V^2 + K \cdot V)$	$O(V^2 \cdot K^2)$

CHAPTER VI

A CASE STUDY

A portion of the flight testing was performed in conjunction with the Defense Advanced Research Project Agency's (DARPA) Heterogeneous Urban Reconnaissance, Surveillance, and Target Acquisition (RSTA) Team (HURT) program. The path planning component that generates waypoints for this case study is closely tied to the one Path Planner (TOPP) which is responsible for assigning sectors to the individual UAVs. Both modules are described in this chapter.

6.1 Problem Description

The system developed for the case study was designed to accommodate the following problem description. The operations involve teams of up to 20 UAVs in some combination of real-world and simulated agents. The UAVs are required to perform missions over a relatively small airspace of 0.5 to 3.5 square miles with an altitude ceiling of 1,000 feet. The ROI may contain numerous no-fly-zones (NFZs) which are to be avoided by the UAVs. The altitude of an individual UAV might be changed during the mission in response to a reconnaissance request. Altitude is the primary deconfliction mechanism among the UAVs.

During the mission, each UAV must be able to operate safely, even if it loses communication with the ground station. The vehicles are capable of flying to and loitering at a rally point when the communication signal is lost. Therefore, the rally point must be placed in a location such that the vehicle can fly to the rally point

without conflicting with any other vehicle or NFZ. The path planner is responsible for determining the proper location of the rally points during the mission.

6.2 Path Planning Approach

The approach for path planning is decomposed into two phases, sector generation and path generation. In the sector generation phase, assigned reconnaissance, surveillance, and target acquisition (RSTA) service requests (RSRs) are sent to TOPP from the planning and execution (PLEX) module incrementally. Each time a new RSR is input into TOPP, an attempt is made to generate a set of sectors to fulfill all the RSRs. This phase must be performed very rapidly (in 10-50 ms) to allow PLEX to assign a different vehicle to the RSR in the cases when a feasible set of sectors is not found. If a set of feasible sectors is generated, the path planner can move into the path generation phase at the request of PLEX. In this phase, a set of waypoints is generated that will allow the vehicle to move in the airspace while staying within its assigned sector.

6.3 Sector Generation

The sector generation approach is summarized in Figure 33. A set of assigned surveillance requests is first input into the coarse RSR route generator. Here, a set of anchor points is generated that will allow the air space to be allocated for the execution of the requested surveillance task. For example, if a request for broad area surveillance is input, a set of anchor points is generated that forms the vertices of a polygon describing the coverage area. These anchor points are intended to form the waypoints of a rough path, which will then drive sector generation.

The anchor points are then sent to the path manager. The path manager is a state

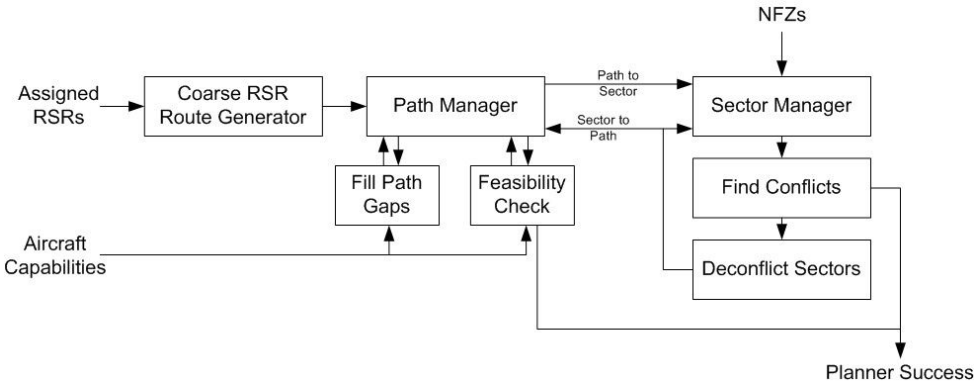

Figure 33: Overview of the sector generator

holding system which keeps track of the current path knowledge. It also manages the assignment of paths to aircraft.

Each time the airspace constraints change, either through a new RSR or NFZ, the path manager sends all the paths for each aircraft to the fill path gaps module. In this module, path segments are inserted to ensure each aircraft has a continuous path. These inserted paths are parameterized by two anchor points, a starting point and an ending point.

After a temporally continuous path is generated for each vehicle, the path is sent through a feasibility check. Here, each path segment is checked to ensure that the ending point of one path is the same as the starting point of the next. After this check, the path segments are checked to ensure that flying from the starting point to the ending point will not require the aircraft to go faster than its specified maximum speed. If a violation is detected, the time alloted for the path segment will be increased if possible. If it is not possible, the RSR that caused the creation of the path will be canceled.

After a continuous, feasible path is generated for each aircraft, sectors are generated for each portion of the path. The sectors are generated by placing a rectangle around the path and then adding a safety margin to the rectangles. These sectors are then sent to the sector manager which is similar to the path manager. However, unlike the path manager the sector manager also keeps track of NFZs.

After all of the sectors are generated, they are checked for conflicts. If a conflict is found, the two conflicting sectors are sent to the deconflict sectors module. The deconfliction is performed by designating one of the sectors as the *mover*, while the other sector is designated as the pusher. First, the mover is examined to determine if its anchor points are on opposite sides of the pusher. If they are, then the sector is resized and moved in the orthogonal direction. For example, if the mover sector is created to move an aircraft from the east side of the pusher sector to the west side of the pusher sector then the mover is resized and moved in the north or south dimension depending on which side is closest. If this resolution results in a conflict with another sector, then the mover is moved in altitude. If this resolution also results in a conflict, then the sector is moved in the opposite direction of the original movement. If this resolution still results in a conflict, the newest RSR is canceled within TOPP and PLEX is notified. Once a sector is moved, its corresponding path is moved through the path manager, and the new, total path is checked for gaps, and continuity.

If the mover sector did not cross the pusher entirely, then a different set of deconfliction rules is used. First, a local grid is generated around the pusher sector as shown in Figure 34 to form candidate regions for possible movements of the mover sector. Then, the angle between the pusher and mover sectors is computed. This angle is used to determine the order in which the candidate regions are searched for any

additional conflicts. For example, if the mover region is to the north of the pusher region then the north candidate region will be searched first, followed the region overhead or below, then the south. After the order is determined, the mover sector is tested in each candidate region following the previously determined search order. As soon as a region is identified where no other conflicts exist, the mover sector is moved to the open region. If no open regions are found, the newest RSR is canceled within TOPP and PLEX is notified. Once a sector is moved, its corresponding path is moved through the path manager, and the new, total path is checked for gaps, and continuity.

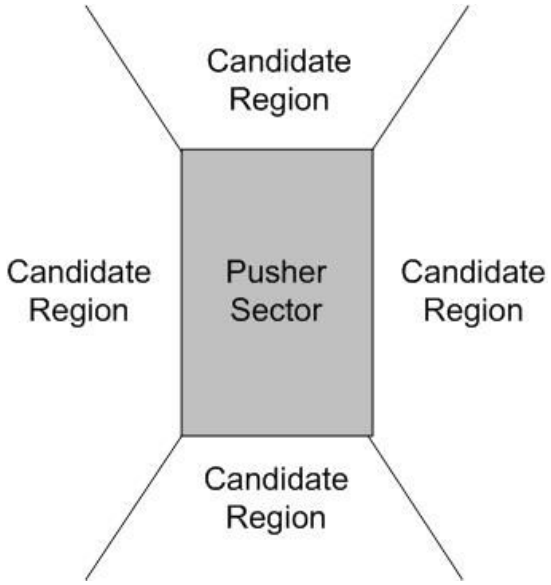

Figure 34: Local grid generated for deconfliction

After sector generation, TOPP returns whether the RSR was inserted or not. TOPP can then provide further path information to PLEX such as, how long will it take for an aircraft to travel to a specific RSR.

6.4 Broad Area Surveillance Assignment

When PLEX sends an assigned broad area surveillance RSR, the sector generator of TOPP also returns an estimate of the coverage rate that would result if the assigned vehicles were used to accomplish the RSR. The estimate is based on the aircraft's speed and sensor footprint. Since, the entire coverage path is not yet generated, the estimate will be some fraction of the nominal, level flight coverage rate. PLEX can use this information to decide if more vehicles need to be assigned to the RSR, or if the user's requirements have been met.

When assigning vehicles to an altitude for broad area surveillance, TOPP will try to keep vehicles at their current altitude. If the current altitude will not meet the RSRs resolution requirements, or if the portion of the altitude layer is occupied by NFZs, TOPP will search for the nearest available altitudes. If the portion of the current altitude layer is occupied by another vehicle servicing the same broad area surveillance RSR and all of the other altitude layers are occupied, TOPP will place multiple vehicles in the same altitude layer.

6.5 Path Generation

After a set of sectors has been assigned for each aircraft, waypoints must be generated to form the path an aircraft is to fly. For these operations it is assumed that three types of paths are required, loiter, point-to-point, and broad area surveillance. Each of these path-types are constrained by their entry point, their exit point, and the sector specified for the aircraft.

The loiter path is anticipated to be used when observing a stationary ground target from a rotary-winged vehicle and as a holding pattern when a vehicle is not

tasked. The path is specified by the sector generator using the same entry and exit points, and a sector that is spatially small. For rotary-winged vehicles, the path will consist of a single waypoint where the vehicle is to hover. For fixed-winged vehicles, the path will consist of a single, loiter type waypoint.

The point-to-point path is anticipated to be used to get a vehicle from one point to another. The sector generator will specify the entry point and the exit point, which will become the first and last waypoint of the path. The location middle points of the path are based on the aircraft capabilities. However, in most cases it is anticipated that the path will simply consist of the two waypoints.

6.6 Waypoint Patterns

It is desired to utilize template patterns in the case study for their predictability and calculation efficiency. The templates include single pass, double pass, lawnmower, spiral, and circumference patterns.

6.7 Multi-Vehicle Coverage Planner

One goal of the case study is for a UAV or a team of UAVs equipped with video cameras to perform visual surveillance over a region of interest. It is assumed that the PLEX or TOPP modules will process the Broad Area Surveillance RSR and assign vehicles to a given altitude and sector in which the vehicles may operate. Then, the Multi-Vehicle Coverage Planner (MVCP) generates a series of waypoints for each vehicle to efficiently view the entire region of interest. The generated waypoints should ensure complete coverage of the region. If a single UAV is assigned to the task then the resulting path will cover the entire region. If multiple vehicles are assigned then the resulting paths should demonstrate co-operative behavior among

the vehicles. An additional goal is to minimize turning in the path generation so that the resulting images can be more easily incorporated into an image montage.

6.7.1 Pattern-Based Approach

The approach used for generating vehicle trajectories leverages template patterns to reduce the computational overhead of the MVCP. These patterns correspond to the desired path of the camera footprint along the ground terrain. The system is extensible to accommodate many pattern templates, however a few patterns proved to be practical. As shown in Figure 35, the lawnmower pattern is simply a repeated out-and-back configuration easily associated with mowing one's lawn and the perimeter pattern follows the edge of the region of interest. The pattern templates are designed to scale with the dimensions of the assigned sector as well as the camera parameters of the UAV.

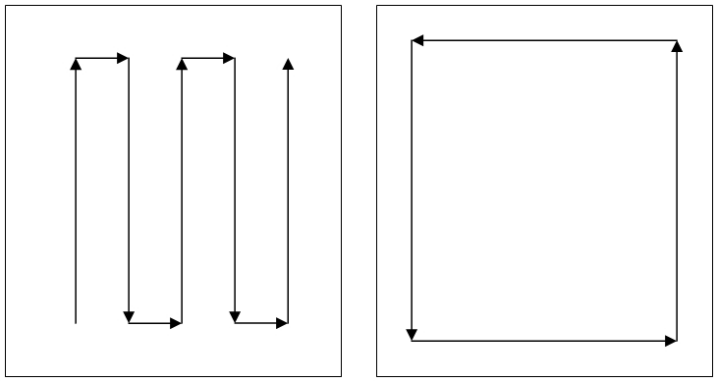

Figure 35: A typical lawnmower pattern and a perimeter pattern.

The path generated for a given vehicle is dependent on the capabilities of all the vehicles that are assigned to the task. UAVs with side-mounted video cameras are preferably assigned a perimeter pattern. The MVCP maintains a snapshot of which

areas have or have not been assigned to coverage. If a perimeter pattern is assigned to a vehicle then the area which will be covered by this pattern is subtracted from the ROI. The last vehicle to be assigned a pattern will always fly a lawnmower pattern to ensure complete coverage.

The previously mentioned patterns are not suitable for small area coverage, however. For small regions (where small is relative to the size of the camera footprint) a straight line or a two-pass pattern is used to view the area with a UAV equipped with a forward facing camera. These patterns assure that a vehicle travels far enough away to guarantee viewing the desired region on the return pass.

After the ground pattern is established for each vehicle, a reverse transformation is performed to generate the waypoints that the UAV must follow to maintain the camera footprint in the desired ground location. On occasion, the ideal flight path is blocked by a no-fly zone (NFZ) or is outside of the vehicle's assigned sector. Therefore, a final check ensures that all of the waypoints are located within the allowed flight space.

Results from the flight tests are reviewed in Chapter 7.

CHAPTER VII

RESULTS

The system described in the previous chapters was flight tested in a case study. Results of that case study led to recommendations for improvement in the overall system design. These improvements led to a new system based on minimum spanning trees. Results from the case study and simulation results of the spanning tree system are presented in the following paragraphs.

7.1 Case Study Results

Flight testing was conducted using Aerovironment Pointer, Raven and Wasp UAV platforms, two of which are shown in Figure 36. A total of four UAVs were flown

(a) (b)

Figure 36: (a) Aerovironment Pointer UAV platform. (b) Aerovironment Raven UAV platform.

simultaneously on coverage missions. Up to 16 additional simulated UAVs were added to the mix of real vehicles. Figure 37 displays an early flight test in which a single

Figure 37: A single UAV is commanded to perform surveillance over a small region during the case study flight tests.

UAV was commanded to perform coverage of a single building within a small region. The zig-zag lines indicate the boundaries of the desired coverage area. The waypoints are displayed as the large octagon and the camera's view is projected onto the ground in the center of the image. The UAV performed its coverage function in the mission scenario well.

Figure 38 displays another flight test with three UAVs in the air performing coverage while a fourth UAV is preparing to launch. Again, the coverage region is bounded by the zig-zag line. Since this region was small compared to the size of the camera's projection, only a circumference pattern was assigned to the UAVs. A ground vehicle is also displayed that was used for tracking purposes.

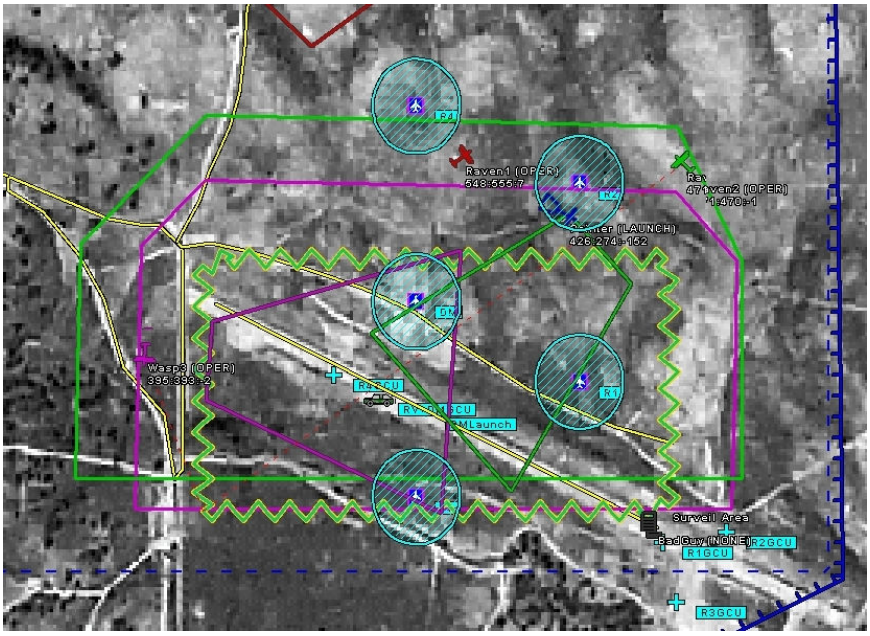

Figure 38: Image from the ground station of live flight tests during the case study.

7.2 MVCP Analysis / Evaluation

During the case study, the MVCP successfully planned routes for multiple vehicles for the broad area surveillance RSRs. In test flights, up to four UAVs were flown simultaneously. The strength of the tile-based system is its low complexity and low computational overhead.

For small regions of interest the system generated appropriate trajectories for the UAVs. For larger areas the generated paths ensured complete coverage of the desired area. The orientation of the generated paths was chosen along the longer axis of the region which resulted in fewer number of turns.

7.3 Simulation Results

After successful flight testing with the template based system, numerous simulations were performed to test the effectiveness of the spanning tree-based system. The ROI was simulated at various sizes roughly equivalent to $1km^2$, $3km^2$, and $10km^2$. The remainder of this section refers to these dimensions simply as a small, medium, or large ROI. The UAVs flew at a simulated velocity of $15m/s$. The region segmentation process results in a graph structure that is abstracted away from dimension units. Therefore, the number of nodes in the graph is also utilized as a metric relating to system performance.

7.4 Coverage Time

The amount of time needed for complete coverage is governed by the maximum distance between UAVs along the flight path. Figure 39 displays the average maximum distance as the number of UAVs increases for a large ROI. As the number of UAVs increases, the likelihood that one or more of the UAVs will become boxed in as the planning routine runs also increases. There are twenty-five simulations for every value of K, where K represents the number of UAVs and $K \in \{1, 2, 3, 4, 5, 10, 15, 20, 25\}$. The graph on the left displays the variation in average distance between UAVs for randomized initial positions of the UAVs. On the right, the graph shows the results of an initially randomized start position with the protective positioning routine activated. It is clear that the variation of the distance is reduced by this procedure.

Achieving equal spacing between the UAVs minimizes the amount of time that a given location in the ROI is not actively viewed. Figure 40 shows the average path length between UAVs for a large ROI as the number of UAVs is swept from 1 to 25.

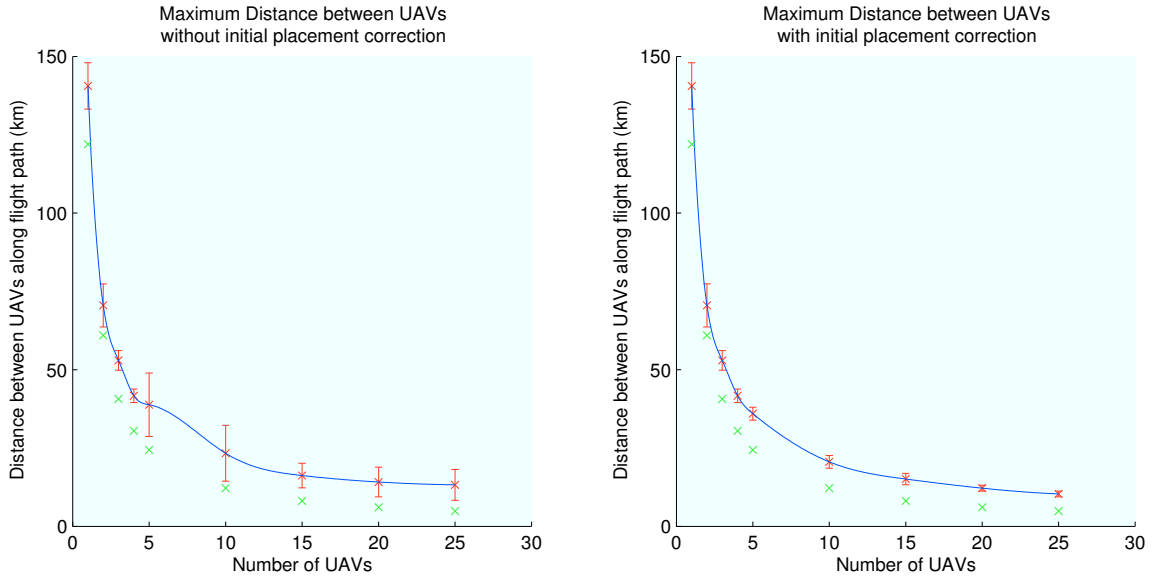

Figure 39: The left graph displays the additional variation in the max distance between UAVs when one of the UAVs is cornered during planning. The right graph displays the distances after initial placement correction is applied.

The green markings indicate a perfect path that include no additional turns. The variation in path length is larger for a single UAV than for more UAVs. The error bars indicate plus or minus one standard deviation. Figure 41 displays the same information for a medium sized ROI.

Perhaps more important than the average value is the maximum value of the UAV to UAV distance for a given test run. For the multi-UAV case, the total coverage time is governed by the vehicle that must travel the farthest distance. The entire region is not completely viewed until the longest distance between UAVs has been traversed. Therefore, the curve is shown again with the maximum values instead of the average values. It can be seen that there is greater variation in the maximum values, which is expected.

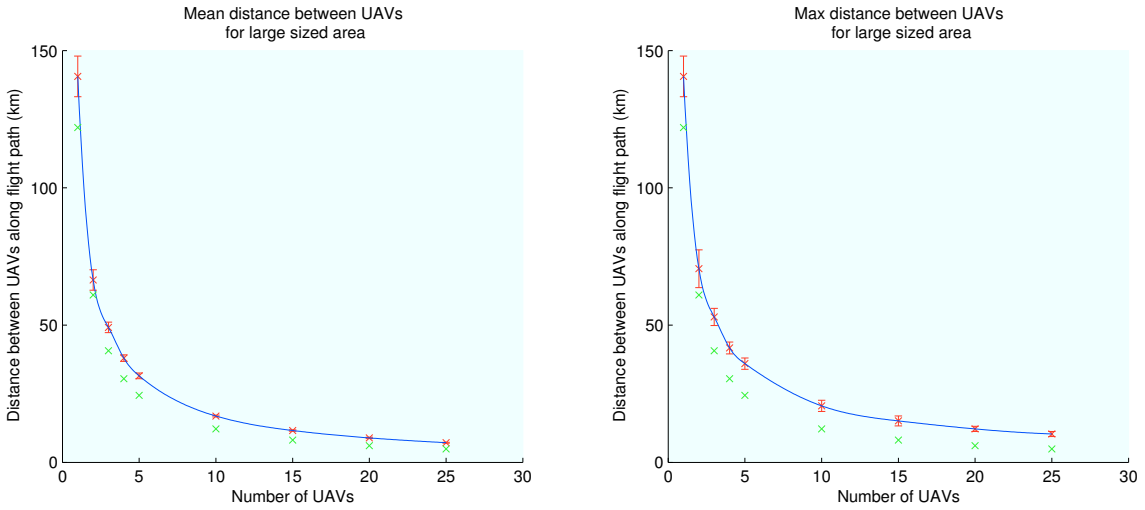

Figure 40: Results of individual path length for various numbers of UAVs over a 1600 node graph.

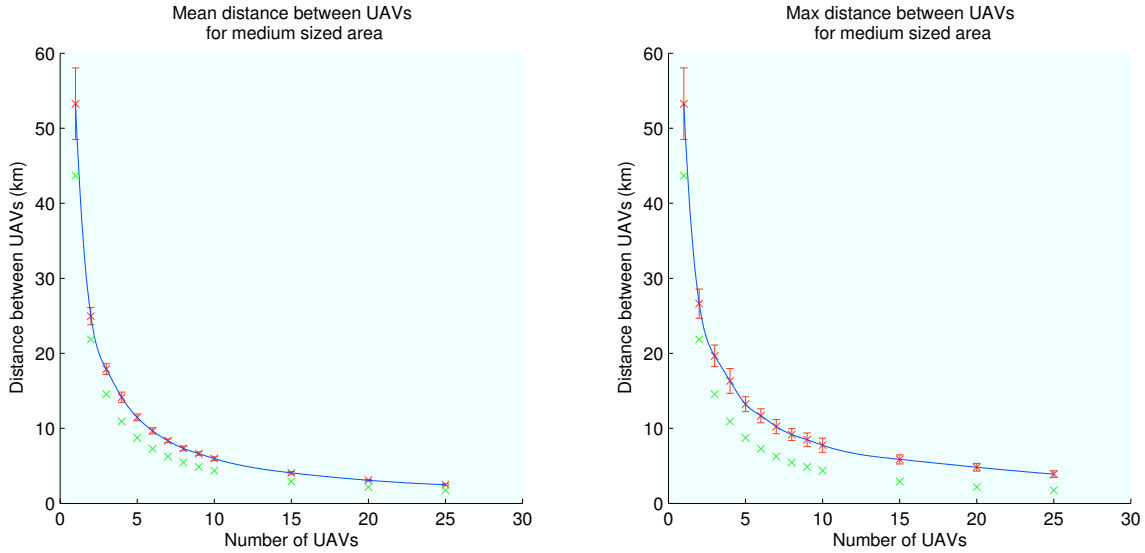

Figure 41: Results of individual path length for various numbers of UAVs over a 500 node graph.

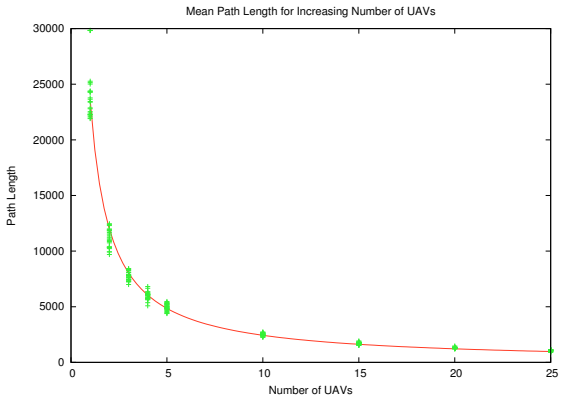

Figure 42: Results of individual path length for various numbers of UAVs over a 200 node graph.

Figure 42 displays the mean distance along the path from UAV_i to UAV_j for adjacent UAVs. The ROI is represented by a $1km^2$ rectangle without any obstacles present. There are about 200 graph nodes for this simulation. There is greater variation for the case of a single UAV than when there are more UAVs in the simulation. The curve displayed in the graph clearly shows the performance benefit of adding additional UAVs. However, it also highlights a point of diminishing returns where the additional UAVs do not significantly improve performance.

The simulation engine shown in Figure 43 was used to set up a surveillance scenario. Two tanks were stationed around an artificial urban setting. One tank was set in motion following the roads, while the other tank remained at rest. All of the simulated UAVs operated at a height of 70 meters from the ground. The user selects the number of UAVs to operate over the region and several time quantities were measured. The time to first contact of either tank and the time required to complete surveillance of the entire region was measured for a varying number of UAVs.

The results of the time to capture simulation are displayed in Figure 44. The time

Figure 43: Depiction of a UAV and a tank in an urban environment as seen during visualization of the simulation.

required to locate a target within the region is a random distribution with an upper bound equal to the maximum time required to achieve complete surveillance.

A different visualization is displayed in Figure 45 that displays more detailed information about the spanning trees, paths, and camera projections. The UAV, its camera projection, and its respective sub-tree share the same color.

7.4.1 Computation Time

Chapter 5 details the complexity analysis of this algorithm. Figure 46 and 47 display measured values of computation time. For these graphs the number of nodes was held constant while the number of UAVs was increased. With the exception of the single UAV case, it can be seen that the computation time increases approximately linearly with the number of UAVs. These results agree with the complexity analysis from Chapter 5. For the largest test cases, with 25 UAVs and a 1600 node graph, the

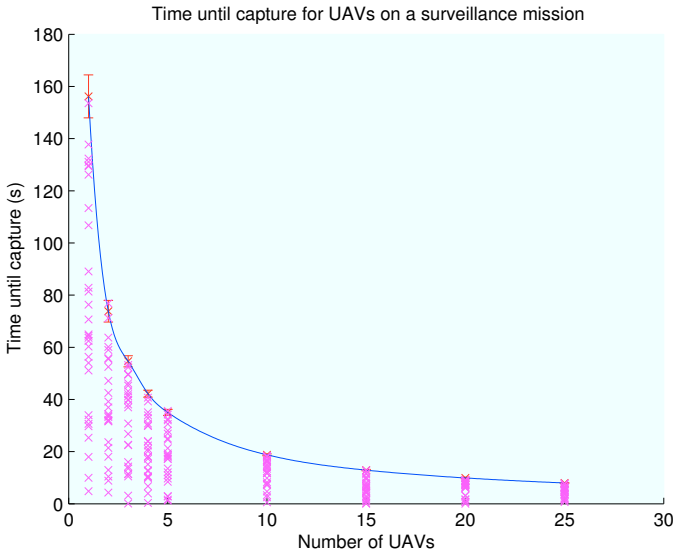

Figure 44: The time to capture a target is a random distribution bounded by the maximum time to complete coverage.

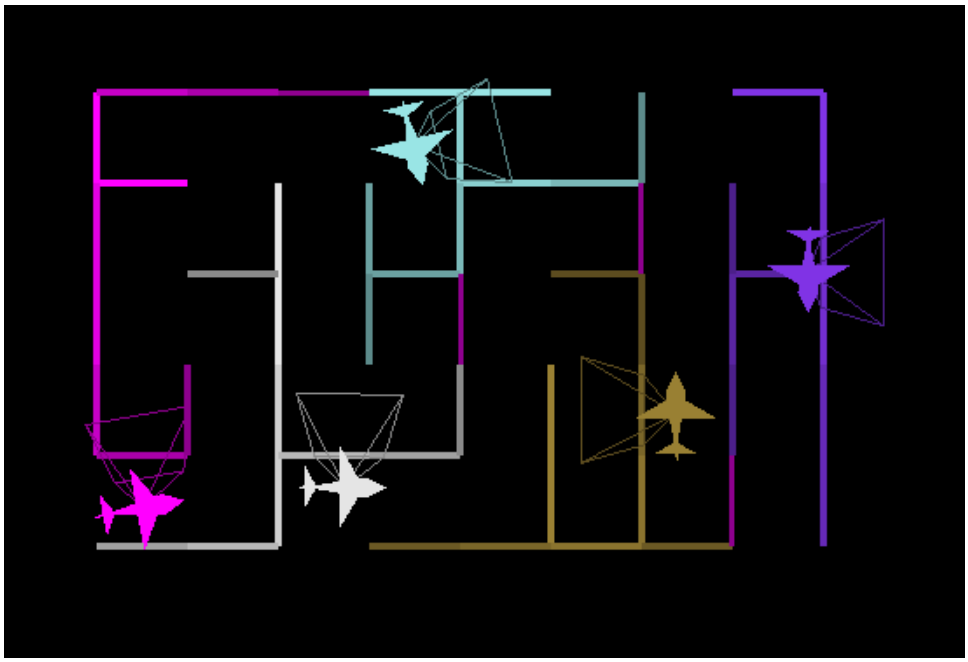

Figure 45: Simulation of five UAVs over a small area showing the color coded spanning trees, bridge connections, and camera projections.

algorithm ran in about 10 seconds. For a test cases on medium or small ROIs, the planning portion required one second or less to compute all of the waypoints.

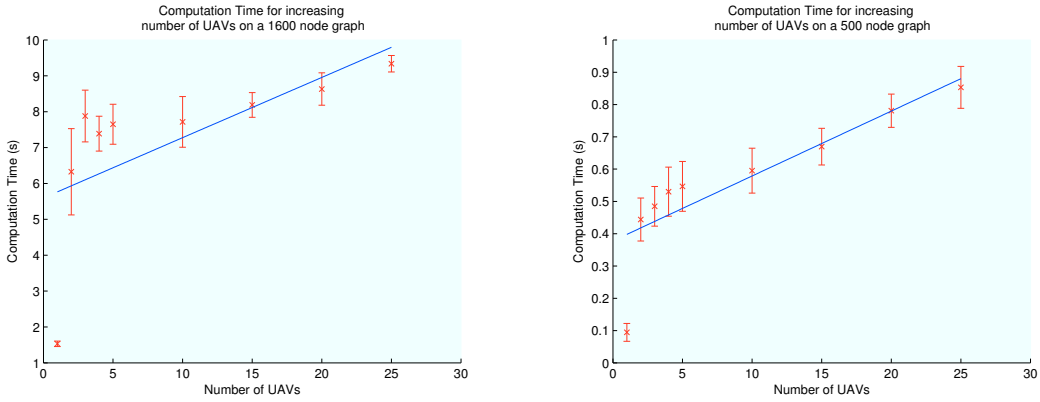

Figure 46: Average UAV to UAV distance along the path for varying numbers of UAVs over a 200 node graph.

Figure 47 displays the number of function calls as the size of the ROI is increased for a fixed number of UAVs. The result is a second order polynomial, which also agrees with the complexity analysis from Chapter 5.

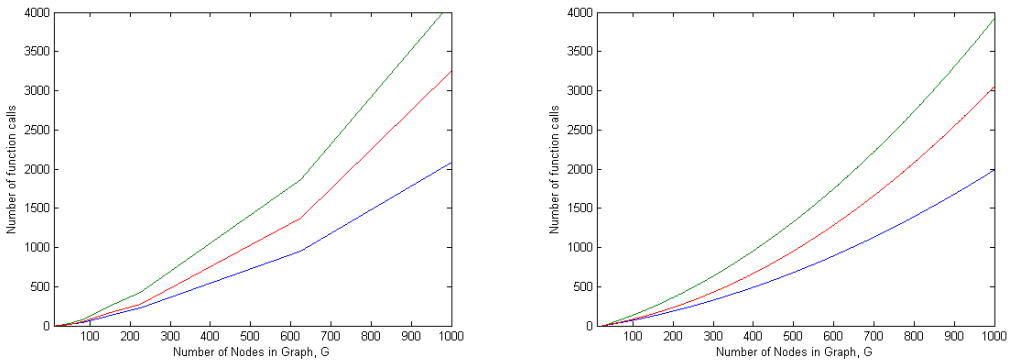

Figure 47: The graph on the left shows the measured execution time with increasing number of vertices and the graph on the right is a second order polynomial approximation of the data on the left.

CHAPTER VIII

CONCLUSIONS AND FUTURE RESEARCH

There is a need to increase the functionality of UAVs so as to reduce the operational burden of controlling UAVs for both military and civilian organizations. The research described here provides tools which facilitate operating a large number of UAVs. The system described in this document efficiently generates waypoints to be followed by UAVs performing surveillance over a large ROI.

This system is based around a minimal spanning tree algorithm, which has been proven useful in a wide variety of applications. The work here builds from a simple ground path planning tool and achieves a multi-vehicle path planner that is efficient, robust, and complete. The algorithm presented maintains a polynomial degree running time in practice despite the problem being NP-hard in general. The path planning portion is general in that it operates entirely on graph structures. Therefore, it may stand alone as an independent tool to work in conjunction with other region segmentation systems.

The simulation results show that UAVs were consistently evenly spaced out along the tour of the ROI. This ensures maximal revisit rate and minimum time between visits for any given point of the ROI. In simulation, the system demonstrated robustness in that if a vehicle failed, the other vehicles would continue to perform surveillance of the ROI, even in the absence of communication. The initial placement of the UAVs ensures that none of the UAVs are blocked into a corner during the planning stage.

The running time of the algorithm presented in this document was in the worst case approximately 10 seconds. This run time compares favorably to other methods in the literature, many of which require hours of computation time.

The parameterized templates were proven in flight testing during the case study. The paths generated by the templates are not as robust against failure, but they are efficient to calculate. Templates are well suited to very small aircraft with limited computational power.

8.1 Contributions

This dissertation contributed to the field by providing the following:

- A control structure that coordinates multiple aerial vehicles in real-time for an area coverage surveillance task.
- A method for initial placement of multiple aerial vehicles for area surveillance.
- An extension of ground-based planning and control techniques to aerial vehicles.
- A fast waypoint generator for multiple unmanned aerial vehicles to perform surveillance.
- A novel method for constructing and combining disjoint balanced spanning trees.

8.2 Future Research

As is frequently encountered, there are areas of this work that could be expanded upon. Occlusion, deconfliction, and refueling are all issues that could improve the overall system with further research.

8.2.1 Wind

Wind has the potential to cause an offset between the desired coverage region versus the region actually viewed by the UAV's camera, as shown in Figure 48. The first UAV depicts the desired result while the second UAV demonstrates the potential camera offset due to wind. The research presented in this document is modular such that the planning up to generating the ground path would remain valid even in high wind conditions. Only the final ground to aerial waypoint transformation would need modification to close the loop to account for wind variations.

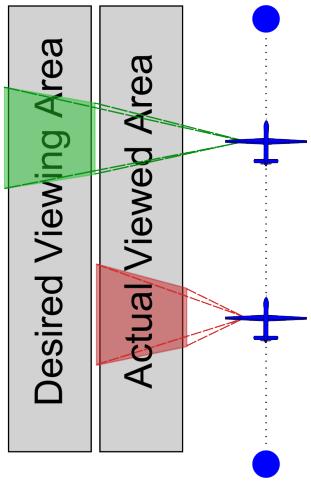

Figure 48: The UAVs may need to adjust for wind conditions during the surveillance mission.

8.2.2 Occlusion

Tall obstacles, such as buildings, and terrain irregularities have the potential to create blind spots during coverage. An urban environment especially creates surveillance challenges. It is apparent that a small UAV is not able to view through a large building in an urban setting. If the terrain data is already known, then the spanning

tree can be modified to account for likely occlusions. In the urban environment, effective techniques need to be examined such as re-using the same ground path but reversing the direction of flight. This would allow the UAVs to view each location from a different viewpoint and significantly reduce the number of occlusions. Other cases of occlusion can be addressed by generating the spanning tree over a finer grid, or offsetting the same sized grid by one-half of the grid spacing.

8.2.3 Deconfliction

Placing a large number of UAVs within a tightly confined region increases the chances that UAVs will crash into each other. This research assumes that the UAVs will be deconflicted by flying at different altitudes. However, a method that allows UAVs to fly at the same altitude level while continuing to avoid collisions would allow a higher concentration of vehicles in the ROI. Further, such a deconfliction system has far reaching implications into improving the safety of airspace management around commercial airports.

8.2.4 Refueling

One major advantage for using multiple UAVs is that they are able to remain aloft for extended periods of time. A human pilot eventually gets tired, whereas a computer does not. However, the aircraft, human piloted or not, will eventually run out of fuel, and it is necessary to address this issue. Ideally, the refueling process can be accommodated without interrupting the surveillance mission, or at least with only minimal interruption. In one technique, the search area has at least one long straight section, preferably at a border of the ROI. The straight section can be utilized by the vehicles as a refueling region in which surveillance is not broken.

Alternatively, there can be a designated refueling area outside the ROI. When a UAV is low on fuel it leaves the ROI and proceeds to the refueling area. After refueling, the UAV proceeds back to its location along the coverage path. In this case, surveillance is only minimally impacted because the remaining vehicles will continue to cover the ROI.

APPENDIX A

DETERMINATION OF LINE SEGMENT INTERSECTION

Generating a path that circumnavigates a spanning tree requires repeatedly testing for the intersection of two lines. The following section details an efficient calculation that determines if two line segments are parallel, intersect within the bounds of the line segments, or intersect outside of the line segments.

Let P_a represent a point on line segment $\overline{P_1P_2}$, and let P_b represent a point on line segment $\overline{P_3P_4}$, as pictured in Figure 49. The equations for P_a and P_b are shown

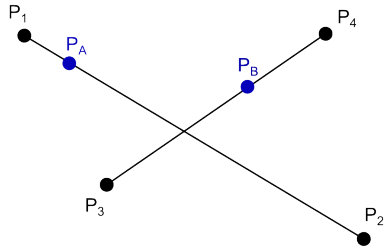

Figure 49: Two intersecting line segments with endpoints labeled.

in Equation 24 and Equation 25.

$$P_a = P_1 + A \cdot (P_2 - P_1) \tag{24}$$

$$P_b = P_3 + B \cdot (P_4 - P_3) \tag{25}$$

A and B are normalized such that $0 \leq A \leq 1$ and $0 \leq B \leq 1$. If the two line segments intersect, then the point of intersection occurs where P_a and P_b are equal. Equating these parameters yields Equation 26.

$$P_1 + A \cdot (P_2 - P_1) = P_3 + B \cdot (P_4 - P_3) \tag{26}$$

Since any point P_i has both x and y co-ordinates, this leads to two equations in x and y.

$$x_1 + A \cdot (x_2 - x_1) = x_3 + B \cdot (x_4 - x_3) \tag{27}$$

$$y_1 + A \cdot (y_2 - y_1) = y_3 + B \cdot (y_4 - y_3) \tag{28}$$

Rearranging these terms leaves A and B as follows:

$$A = \frac{(x_3 - x_1) + B \cdot (x_4 - x_2)}{(x_2 - x_1)} \tag{29}$$

$$B = \frac{(x_1 - x_3) + A \cdot (x_2 - x_1)}{(x_4 - x_3)} \tag{30}$$

Finally, A and B can be solved strictly in terms of the x and y components of the original set of points (P_1, P_2, P_3, P_4).

$$A = \frac{(x_1 - x_3)(y_4 - y_3) - (y_1 - y_3)(x_4 - x_3)}{(y_2 - y_1)(x_4 - x_3) - (x_2 - x_1)(y_4 - y_3)} \tag{31}$$

$$B = \frac{(y_2 - y_1)(x_1 - x_3) - (x_2 - x_1)(y_1 - y_3)}{(y_2 - y_1)(x_4 - x_3) - (x_2 - x_1)(y_4 - y_3)} \tag{32}$$

Several common term can be collected to reduce the number of calculations needed to find A and B. The denominator is identical for both variables. The values of A and B are then tested to determine the orientation of the two lines with respect to each other. For a given two line segments, if $0 \leq A \leq 1$ and $0 \leq B \leq 1$ then the line segments do intersect. If the denominator in the above equations is equal to zero,

then the line segments are parallel and do not intersect. If $A > 1$ or $B > 1$ then the extended lines intersect at a location exterior to the line segments.

APPENDIX B

DISTANCE BETWEEN UAVS ALONG A PATH THAT CIRCUMNAVIGATES MULTIPLE SPANNING TREES

Let K represent the number of individual spanning trees. Let $D(T_i)$ represent the linear distance of a path, P_i, that circumnavigates the i^{th} tree. Every spanning tree, T_i, has an associated UAV, UAV_i, whose position is known at some point along the path circumnavigating T_i. The goal is to find a set of $K-1$ edges that connects all the trees together such that the UAVs are evenly dispersed along the path that circumnavigates the resultant single spanning tree. The sum of the distances around each sub-section of the tree is equal to the total distance around the combined trees.

$$\sum_{i=1}^{K} D(T_i) = D_{total} \tag{33}$$

To find the general form, examine the set of $K \in (2, 3, ..., N)$. The directed spanning tree method described earlier in this document generates a set of trees that are nearly equal in their number of vertices and edges. Equation 34 shows the ideal case for $K = 2$.

$$D(T_1) = D(T_2) = \frac{D_{total}}{2} \tag{34}$$

In practice, the length of the distance around each sub-tree is different, that is,

$D(T_1) \neq D(T_2)$. Let α_i represent the error distance in sub-tree i such that

$$D(T_1) = \frac{D_{total}}{2} + \alpha_1 \tag{35}$$

$$D(T_2) = \frac{D_{total}}{2} + \alpha_2 \tag{36}$$

there exists an error, ϵ, such that . Instead, Equation 37 represents the distance of the path around a single tree including the likely error, ϵ

$$D(T_i) = \frac{D_{total}}{K} + \epsilon_i \tag{37}$$

For $K = 2$ there are two potential cases: the distance from UAV 1 to UAV 2 is greater than the distance from UAV 2 to UAV 1, or the opposite.

It is known from Equation 33 that the α terms must sum to zero.

$$\sum_{i=1}^{K} \alpha_i = 0 \tag{38}$$

Therefore, for this case the two α terms are equal in magnitude with opposite sign.

$$\alpha_1 = -\alpha_2 \tag{39}$$

There exists one UAV per sub-tree. Nominally the distance along the path from UAV_1 to UAV_2 is equal to the distance along the path from UAV_2 to UAV_1. However, in practice there exists some error, $\epsilon_{i,j}$, between the equi-distant path and the actual

path from UAV_i to UAV_j.

$$D_{1,2} = \left(\frac{D(T_1) + D(T_2)}{2}\right) + \epsilon_{1,2} \tag{40}$$

$$= \frac{D_{total}}{K} + \left(\frac{\alpha_1 + \alpha_2}{2}\right) + \epsilon_{1,2} \tag{41}$$

$$= \frac{D_{total}}{K} + \epsilon_{1,2} \tag{42}$$

$$D_{2,1} = \left(\frac{D(T_2) + D(T_1)}{2}\right) + \epsilon_{2,1} \tag{43}$$

$$= \frac{D_{total}}{K} + \left(\frac{\alpha_2 + \alpha_1}{2}\right) + \epsilon_{2,1} \tag{44}$$

$$= \frac{D_{total}}{K} + \epsilon_{2,1} \tag{45}$$

This is written more generally as:

$$D_{i,j} = \frac{D_{total}}{K} + \epsilon_{i,j} \tag{46}$$

For $K = 3$ there exist only one mathematically distinct case for the way in which the bridges connect the individual sub-trees. Once again, the total distance is given by the summation of distances.

$$\sum_{i=1}^{K} D(T_i) = D_{total} \tag{47}$$

The length of each sub-tree is given by the average distance plus an error term, α.

$$D(T_i) = \frac{D_{total}}{K} + \alpha_i \tag{48}$$

$$D_{1,2} = \left(\frac{D(T_1) + D(T_2)}{2}\right) + \epsilon_{1,2} \tag{49}$$

$$= \frac{D_{total}}{K} + \left(\frac{\alpha_1 + \alpha_2}{2} + \epsilon_{1,2}\right) \tag{50}$$

$$D_{2,3} = \left(\frac{D(T_2) + D(T_3)}{2}\right) + \epsilon_{2,3} \tag{51}$$

$$= \frac{D_{total}}{K} + \left(\frac{\alpha_2 + \alpha_3}{2} + \epsilon_{2,3}\right) \tag{52}$$

$$D_{3,1} = D_{total} - (D_{1,2} + D_{2,3}) \tag{53}$$

$$= \frac{D_{total}}{K} + \left(\frac{\alpha_3 + \alpha_1}{2} - (\epsilon_{1,2} + \epsilon_{2,3})\right) \tag{54}$$

More general, the distance from UAV_i to UAV_j along the path can be written as follows.

$$D_{i,j} = \frac{D_{total}}{K} + \left(\frac{\alpha_i + \alpha_j}{2} + \epsilon_{i,j}\right) \tag{55}$$

In fact, Equation 46 is also of the same form as Equation 55. However, for $K = 2$ $\alpha_i + \alpha_j = 0$.

Table 3 lists the possible orders of the UAVs along a tour that encompasses the spanning tree with four UAVs.

Table 3: Possible order of UAVs along a path with $K = 4$

Case 1:	$UAV_1 \rightarrow UAV_2 \rightarrow UAV_3 \rightarrow UAV_4$
Case 2:	$UAV_1 \rightarrow UAV_3 \rightarrow UAV_4 \rightarrow UAV_2$
Case 3:	$UAV_1 \rightarrow UAV_4 \rightarrow UAV_3 \rightarrow UAV_2$
Case 4:	$UAV_1 \rightarrow UAV_2 \rightarrow UAV_4 \rightarrow UAV_3$

Although there exist four possible orders, only two are mathematically distinct methods of connecting the sub-trees. The first of these is as follows:

$$D_{1,2} = \left(\frac{D(T_1) + D(T_2)}{2}\right) + \epsilon_{1,2} \tag{56}$$

$$= \frac{D_{total}}{K} + \left(\frac{\alpha_1 + \alpha_2}{2} + \epsilon_{1,2}\right) \tag{57}$$

$$D_{2,3} = \left(\frac{D(T_2) + D(T_3)}{2}\right) + \epsilon_{2,3} \tag{58}$$

$$= \frac{D_{total}}{K} + \left(\frac{\alpha_2 + \alpha_3}{2} + \epsilon_{2,3}\right) \tag{59}$$

$$D_{3,4} = \left(\frac{D(T_3) + D(T_4)}{2}\right) + \epsilon_{3,4} \tag{60}$$

$$= \frac{D_{total}}{K} + \left(\frac{\alpha_3 + \alpha_4}{2} + \epsilon_{3,4}\right) \tag{61}$$

$$D_{4,1} = D_{total} - (D_{1,2} + D_{2,3} + D_{3,4}) \tag{62}$$

$$= \frac{D_{total}}{K} - \left(\frac{\alpha_2 + \alpha_3}{2} + \epsilon_{1,2} + \epsilon_{2,3} + \epsilon_{3,4}\right) \tag{63}$$

$$= \frac{D_{total}}{K} + \left(\frac{\alpha_3 + \alpha_1}{2} - (\epsilon_{1,2} + \epsilon_{2,3})\right) \tag{64}$$

In the second mathematically distinct possibility, $D_{1,3}$ is that path that lies adjacent to tree 1, tree 2, and tree 3 but UAV2 happens to lie on the other side of the path. To examine this possibility, look at the sub-set of 3 trees, temporarily leaving off the fourth.

Let

$$D_3 = D(T_1) + D(T_2) + D(T_3) \tag{65}$$

$$= \frac{3 * D_{total}}{K} + (\alpha_1 + \alpha_2 + \alpha_3) \tag{66}$$

From the $K = 3$ analysis it is known that

$$D_{3,2} = \frac{D_{total}}{K} + \left(\frac{\alpha_3 + \alpha_2}{2} + \epsilon_{3,2} \right) \tag{67}$$

$$D_{2,1} = \frac{D_{total}}{K} + \left(\frac{\alpha_2 + \alpha_1}{2} + \epsilon_{2,1} \right) \tag{68}$$

$$D_{1,3} = D_3 - (D_{3,2} + D_{2,1}) \tag{69}$$

$$= \frac{D_{total}}{K} + (\alpha_1 + \alpha_2 + \alpha_3) - \left(\frac{\alpha_1 + \alpha_3}{2} \right) - \alpha_2 - (\epsilon_{3,2} + \epsilon_{2,1}) \tag{70}$$

$$= \frac{D_{total}}{K} + \left(\frac{\alpha_1 + \alpha_3}{2} + \epsilon_{1,3} \right) \tag{71}$$

The equations for $K = 4$ and higher can be reduced to forms already analyzed. Therefore, the general form is shown in Equation 72

$$D_{i,j} = \frac{D_{total}}{K} + \left(\frac{\alpha_i + \alpha_j}{2} + \epsilon_{i,j} \right) \tag{72}$$

REFERENCES

[1] Agarwal, A., Hiot, L. M., Joo, E. M., and Nghia, N. T., "Rectilinear workspace partitioning for parallel coverage using multiple uavs," *Advanced Robotics*, vol. 21, no. 1, pp. 105–120, 2006.

[2] Agmon, N., Hazon, N., and Kaminka, G. A., "Constructing spanning trees for efficient multi-robot coverage," *IEEE International Conference on Robotics and Automation*, May 2006.

[3] Ahmadi, M. and Stone, P., "A multi-robot system for continuous sweeping tasks," in *2006 IEEE International Conference on Robotics and Automation*, (Orlando, Florida), pp. 1724–1729, IEEE, May 2006.

[4] Ahmadzadeh, A., Keller, J., Pappas, G. J., Jadbabaie, A., and Kumar, V., "Multi-uav cooperative surveillance with spatio-temporal specifications," *Proceedings of the IEEE Conference on Decisions and Control*, Dec 2006.

[5] Al-Hasan, S. and Vachtsevanos, G., "Intelligent route planning for fast autonomous vehicles operating in a large natural terrain," *Robotics and Autonomous Systems*, no. 40, pp. 1–24, 2002.

[6] Arkin, E. and Refael, H., "Approximation algorithms for the geometric covering salesman problem," *Discrete Applied Mathematics*, vol. 55, pp. 197–218, 1994.

[7] Arkin, E. M., Fekete, S. P., and Mitchell, J. S. B., "Approximation algorithms for lawn mowing and milling," *Computational Geometry*, vol. 17, no. 1-2, pp. 25–50, 2000.

[8] Arkin, E. M., Hassin, R., and Levin, A., "Approximations for minimum and min-max vehicle routing problems," *J. Algorithms*, vol. 59, no. 1, pp. 1–18, 2006.

[9] Bayraktar, S., Fainekos, G. E., and Pappas, G. J., "Experimental cooperative control of fixed-wing unmanned aerial vehicles," *IEEE Conference on Decision and Control*, vol. 4, pp. 4292–4298, Dec 2004.

[10] Carmi, P., Katz, M. J., and Mitchell, J. S., "The minimum-area spanning tree problem," *WADS*, pp. 195–2004, 2005.

[11] Casbeer, D., Beard, R., McLain, T., Li, S.-M., and Mehra, R., "Forest-fire monitoring with multiple small uavs," *American Controls Conference*, pp. 3530–3535, June 2005.

[12] Chizek, J. G., "Military transformation: Intelligence, surveillance and reconnaissance," *Congressional Research Service*, 2003.

[13] Choset, H., "Coverage of known spaces: The boustrophedon cellular decomposition," *Autonomous Robots*, vol. 9, pp. 247–253, 2000.

[14] Choset, H., "Coverage for robotics - a survey of recent results," *Annals of mathematics and artificial intelligence*, vol. 31, pp. 113–126, 2001.

[15] Choset, H. and Pignon, P., "Coverage path planning: the boustrophedon cellular decomposition," *Proceedings of the International Conference on Field and Service Robotics*, December 1997.

[16] Deng, X. and Mirzaian, A., "Competitive robot mapping with homogoneous markers," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 532–542, 1996.

[17] DoD, "Unmanned aerial systems roadmap 2002-2027," tech. rep., Office of the Secretary of Defense, Washington, DC 20301, December 2002.

[18] DoD, "Unmanned aerial systems roadmap 2005-2030," tech. rep., Office of the Secretary of Defense, Washington, DC 20301, August 2005.

[19] Eiben, A., Nitschke, G., and Schut, M., "Evolving an agent collective for cooperative mine sweeping," *Evolutionary Computation. The 2005 IEEE Congress on*, vol. 1, pp. 831–836, Sept 2005.

[20] Even, G., Garg, N., Konemann, J., Ravi, R., and Sinha, A., "Min-max tree covers of graphs," *Operations Research Letters*, no. 31, pp. 309–315, 2004.

[21] Gabriely, Y. and Rimon, E., "Spanning-tree based coverage of continuous areas by a mobile robot," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 77–98, 2001.

[22] Gabriely, Y. and Rimon, E., "Spiral-stc: An on-line coverage algorithm of grid environments by a mobile robot," *IEEE International Conference on Robotics and Automation*, pp. 954–960, May 2002.

[23] Garcia, E. and de Santos, P. G., "Mobile-robot navigation with complete coverage of unstructured environments," *Robotics and Autonomous Systems*, vol. 46, pp. 195–204, Feb 2004.

[24] Graham, R. L. and Hell, P., "On the history of the minimum spanning tree problem," *Annals of the History of Computing*, vol. 7, January 1985.

[25] Grocholsky, B., Bayraktar, S., Kumar, R., Taylor, C. J., and Pappas, G. J., "Synergies in feature localization by air-ground robot teams," *International Symposium on Experimental Robotics, Proceedings of*, June 2004. Papar ID 149.

[26] Grocholsky, B., Keller, J., Kumar, V., and Pappas, G., "Cooperative air and ground surveillance," *IEEE Robotics and Automation Magazine*, vol. 13, pp. 16–25, Sept 2006.

[27] Guo, Y. and Qu, Z., "Coverage control for a mobile robot patrolling a dynamic and uncertain environment," *Proceedings of the 5th World Congress on Intelligent Control and Automation*, pp. 4899–4903, June 2004.

[28] Hazon, N. and Kaminka, G., "Redundancy, efficiency and robustness in multi-robot coverage," *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 735–741, 2005.

[29] Hazon, N., Mieli, F., and Kaminka, G. A., "Towards robust on-line multi-robot coverage," in *2006 International Conference on Robotics and Automation*, 2006.

[30] Hegazy, T., Ludington, B., and Vachtsevanos, G., "Reconnaissance and surveillance in urban terrain with unmanned aerial vehicles," *Proceedings of 16th IFAC World Congress*, July 2005.

[31] Hegazy, T. and Vachtsevanos, G., "Sensor placement for isotropic source localization," *The Second International Workshop on Information Processing in Sensor Networks (ISPN '03)*, April 2003.

[32] Hert, S., Tiwari, S., and Lumelsky, V., "A terrain-covering algorithm for an auv," *Journal of Autonomous Robots*, vol. 3, pp. 91–119, 1996.

[33] Hofner, C. and Schmidt, G., "Path planning and guidance techniques for an autonomous mobile cleaning robot," *Robotics and Autonomous Systems*, vol. 14, pp. 199–212, 1995.

[34] Huang, W., "Optimal line-sweep decompositions for coverage algorithms," *2001 International Conference on Robotics and Automation*, vol. 1, pp. 27–32, 2001.

[35] Huang, Y., Cao, Z., and Hall, E., "Region filling operations for mobile robot using computer graphics," *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, vol. 3, pp. 1607–1614, Apr 1986.

[36] Jager, M. and Nebel, B., "Dynamic decentralized area partitioning for cooperatting cleaning robots," *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, pp. 3577–3582, May 2002.

[37] Kappenman, J., "Unmanned aerial systems in the u.s. army," *Proceedings of the IDGA Summit*, 2006.

[38] Kruskal, J. B., "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, pp. 48–50, Feb 1956.

[39] Kurabayashi, D., Ota, J., Arai, T., and Yoshida, E., "Cooperative sweeping by mutliple mobile robots," in *International Conference on Robotics and Automation*, (Minneapolis, Minnesota), pp. 1744–1749, IEEE, April 1996.

[40] Maza, I. and Ollero, A., "Multiple uav cooperative searching operation using polygon area decomposition and efficient coverage algorithms," *International Symposium on Distributed Autonomous Robotic Systems*, no. 7, pp. 211–220, 2004.

[41] NIEDERMEIER, R. and SANDERS, P., "On the manhattan-distance between points on space-filling mesh-indexings.," Tech. Rep. iratr-1996-18, 1996.

[42] Ore, O., "A note on hamiltonian circuits," *American Mathematical Monthly*, vol. 55, no. 67, 1960.

[43] Prim, R., "Shortest connection networks and some generalisations," Tech. Rep. 36, Bell System Technical Journal, August 1957.

[44] Quigley, M., Barber, B., Griffiths, S., and Goodrich, M. A., "Towards real-world searching with fixed-wing mini-uavs," *IEEE International Conference on Intelligent Robots and Systems*, pp. 3028–3033, August 2005.

[45] Smith, J., Campbell, S., and Morton, J., "Design and implementation of a control algorithm for an autonomous lawnmower," *Circuits and Systems, 2005. 48th Midwest Symposium on*, vol. 1, pp. 456–459, Aug 2005.

[46] Sutton, A., Fidan, B., and van der Walle, D., "Hierarchical uav formation control for cooperative surveillance," in *The 17th World Congress The International Federation of Automatic Control*, (Seoul, Korea), pp. 12087–12092, July 2008.

[47] Tang, Z. and Ozguner, U., "Motion planning for multitarget surveillance with mobile sensor agents," *IEEE Transactions on Robotics*, vol. 21, pp. 898–908, October 2005.

[48] Vachtsevanos, G., Tang, L., Drozeski, G., and Gutierrez, L., "From mission planning to flight control of unmanned aerial vehicles: Strategies and implementation tools," *Annual Reviews in Control*, vol. 29, pp. 101–115, April 2005.

[49] Zheng, X., Jain, S., Koenig, S., and Kempe, D., "Multi-robot forest coverage," *Intelligent Robots and Systems*, pp. 3852–3857, August 2005.

VITA

Phillip J. Jones spent his early years in Memphis, TN and earned his Bachelor of Science in Electrical Engineering from the University of Tennessee, Knoxville. After graduating, he worked as an analog design engineer at Harris Corporation, where he contributed design expertise to the International Space Station and the Comanche RAH-66 Helicopter. Then, he began graduate school at Georgia Tech Lorraine, France. He learned to speak French and earned a Master Degree in Electrical and Computer Engineering as well as a Diplôme D'Ingénieur. His research interests include unmanned aerial vehicles and intelligent control systems.

RELATED PUBLICATIONS

JONES, P., and VACHTSEVANOS, G., "Multi-Unmanned Aerial Vehicle Coverage Planner for Area Surveillance Missions" in *Proceedings of the AIAA Guidance Navigation and Control Conference*, (Hilton Head, SC), 2007.

JONES, P., LUDINGTON, B., REIMANN, J., and VACHTSEVANOS, G., "Intelligent Control of Unmanned Aerial Vehicles for Improved Autonomy," *European Control Conference*, 2007.