Altitude-Constrained Occlusion-Aware UAV Coverage Planning for Autonomous Wilderness Search and Rescue

Daniel J. Broyles

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2023

Reading Committee:

Juris Vagners, Chair

Karen Leung, Chair

Amir Taghvaei

Program Authorized to Offer Degree:
Aeronautics and Astronautics

University of Washington

Abstract

Altitude-Constrained Occlusion-Aware UAV Coverage Planning for Autonomous Wilderness Search and Rescue

Daniel J. Broyles

Co-Chairs of the Supervisory Committee:
Professor Juris Vagners
Department of Aeronautics and Astronautics

Assistant Professor Karen Leung
Department of Aeronautics and Astronautics

Autonomous Unoccupied Aerial Vehicles (UAVs) have transformed how search and rescue (SAR) missions operate, enabling rapid real-time video feedback over large and difficult-to-access areas. However, ensuring full search coverage in wilderness environments is challenging, and current UAV path planning solutions do not account for occlusion caused by complex terrains and dense vegetation. In wilderness settings, UAV operators often rely on manual control to mitigate the effects of occlusion, performing on-the-fly position adjustments to achieve less-occluded viewing angles. To address the challenge of autonomous occlusion-aware coverage path planning, this dissertation presents VWSGA+, a waypoint path planning algorithm that guarantees complete coverage in occluded environments. The practicality and theoretical guarantees of VWSGA+ are demonstrated in a full-scale simulation and real-world experiments, in which the VWSGA+ is compared against state-of-the-art methods and is shown to be superior in providing complete coverage using battery-efficient paths. This dissertation advances the state of autonomy for UAV-assisted wilderness SAR operations by producing high-confidence aerial search coverage paths, ultimately enhancing the search team's ability to provide this life-saving service to society.

The views expressed in this document are those of the author and do not reflect the official policy or position of the United States Air Force, the United States Department of Defense or the United States Government.

TABLE OF CONTENTS

LIST OF FIGURES

GLOSSARY

AGP: Art Gallery Problem

CHM: Canopy Height Model

DTM: Digital Terrain Model

DSM: Digital Surface Model

FOV: Field of View

GCN: Graph Convolutional Network

GCS: Ground Control Station

GNN: Graph Neural Network

ITDS: Individual Tree Detection and Segmentation

SAR: Search and Rescue

SAT: Boolean Satisfiability

SMT: Satisfiability Modulo Theory

TSP: Traveling Salesman Problem

UAV: Unoccupied Aerial Vehicle

VWSGA: Vantage Waypoint Set Generation Algorithm

WISAR: Wilderness Search and Rescue

ACKNOWLEDGMENTS

It is tremendously humbling to think about the debt of gratitude I owe to my friends, family, colleagues, advisors, leaders, and mentors. In very direct and measurable ways, these people guided my transformation from a directionless college dropout into the person I am today. With this latest milestone behind me, I would like to celebrate the people who made this possible with a few toasts:

First, to my co-advisors, Juris and Karen. What a rare gift and honor it was to be the swan song and culmination of an accomplished Emeritus professor's academic journey, while simultaneously witnessing the launch of a promising and exceptionally talented new professor's career. I cannot thank you both enough for your guidance, wisdom, patience, and example, which have not only made me a better academic researcher, but also a better leader and mentor than I was upon starting this program. I would also like to thank my committee, Amir, David, and J.B, for their extremely useful insights and unwavering support for my research every step of the way.

To my colleagues in the Control and Trustworthy Robotics Lab (CTRL), who rallied to help me execute my research plan and donated their time and energies to solving the myriad of problems we encountered along the way. I would like to especially thank Chris, who is the most resourceful and talented researcher I've ever met and was my "battle buddy" throughout my time here - thank you for your friendship, your significant contributions to this research, and for the wonderful memories and experiences we shared, both here and overseas - I could not have done this without you! To Helen, who is an absolute joy to work with and who was my backstop, ensuring our operations were always safe and successful (despite her strange ability to

make electronic systems fail with her mere presence) - thank you for all your support and for keeping a safe distance from my electronic systems!

To my Qualification Exam Study Group; Eddie, Nick, John, Taewan, Carey, Kazu, Aditya and Dan. I am so grateful to have worked with, and learned from, all of you. I credit our little study group as the catalyst which helped me overcome the difficulties with pandemic-era learning and get past the most difficult step in the PhD process. There is no way I would have finished on this accelerated timeline without you!

To my best friend Jim, who helped me achieve far more than I thought was possible, who came to my rescue by elegantly solving a difficult problem I was facing, and who has consistently been a pillar of strength and a beacon of wisdom for me and my family. Thank you for always making it clear that you believed in me - I am a better person in every way for having you in my life.

To the late Reggie Tilman, the savvy Air Force veteran who tricked me into joining the Air Force, a life-changing decision which has widely been considered to be a good move. I am forever grateful to you and miss you dearly, old friend. And to my first Air Force mentor, Dr. Brownie, whom I credit with jump-starting my academic journey, for teaching me the meaning of service before self, and for guiding me to success in an extremely competitive enlisted-to-officer program. It was an absolute privilege to have served with you; your example of servant leadership is still the greatest I have witnessed in my nearly 20 years of service, and it has become the standard I strive for in my own leadership style.

Most importantly, to my amazing family, whose support kept me going. To my mother and father, who never left me wanting for love. To my aunt and uncle, whose example have shaped my understanding of what it means to be a good human being. To my grandfather, whose life inspired me to aim higher, and to my grandma, who is, and always will be, the most wonderful person I have ever known. To brother and

sister-in-law, who are a tremendous source of joy in our lives. To my boys, whose smiles, hugs, and encouragement brightened my toughest days.

And to my wife, who has been my partner in every sense of the word. Words cannot express what your love and sacrifice mean to me, nor the gratitude I have for having you in my life - thank you for standing with me through thick and thin these 18 years, and I look forward to the many more years of love, companionship, and shared adventures together as we continue this journey.

DEDICATION

To Dimples, my one and only and my very best friend.

To my boys, who have made me immeasurably proud, and whose laughter is my favorite sound in the world.

And to Reggie, my mentor and friend, who saw potential in me when I was at my lowest, and guided me onto a better path.

Chapter 1
INTRODUCTION

Around the world, Wilderness Search and Rescue (WiSAR) teams expend an immense amount of human effort and resources every year in order to provide a life-saving service to society. These teams face difficult challenges in navigating and searching through vast areas of rugged and unforgiving terrain in search of lost individuals in remote locations (see Fig. 1.1). In the U.S. alone, the National Park Service reported 4,194 incidents in 2017, resulting in an estimated cost of $2.1M. Of these incidents, over 3.5k resulted in saves (lost subject found and rescued), 187 resulted in fatalities, and 423 lost subjects were, unfortunately, never found [1].

WiSAR missions are time-critical, and the ability to quickly search for and locate a lost subject significantly increases survival and the probability of mission success [2]. Often, searching for and locating the lost individual is the most difficult and time-consuming step in a search and rescue operation. In some cases, helicopters or fixed-wing aircraft are employed, enabling coverage of difficult-to-access regions and the ability to search larger areas in less time than would be possible with ground crews alone. However, most search and rescue organizations lack adequate funding to support aerial search missions and rely heavily on an all-volunteer force [3]. On the other hand, Unoccupied Aerial Vehicles (UAVs) offer a viable, lower-cost alternative to aircraft, and their use for search and rescue missions has grown significantly over the past decade [4].

Figure 1.1: WiSAR Operational Challenges (left to right): remote locations with restricted access, high-risk operations for ground teams, challenging terrain features, small object detection from aerial imagery, often accomplished as an additional duty for the UAV operator.

1.1 UAV-Enabled WiSAR Concept of Operations

Fig. 1.2 depicts the concept of operations for a UAV-enabled WiSAR operation. A typical operation is overseen by a designated Incident Commander, who directs the search effort and allocates resources. The UAV operator's role is to collect georeferenced imagery of areas-of-interest identified by the Incident Commander in a way that supports both the detection task (i.e., determine if the missing person is present in the images) and the search team's decision-making processes (i.e., figuring out how the search should proceed and where to search next). Analysis of the collected imagery is performed in real-time either by a designated (human) imagery analyst or by the UAV operator, with the goal of detecting or identifying evidence of the lost individual's presence or absence.

Ideally, if the imagery supports a positive detection with high-confidence, meaning the lost individual is indeed located within the area-of-interest, the imagery collected will help the search team easily identify the location and condition of the lost subject. If the evidence from the imagery is inconclusive, the Incident Commander may decide to send a ground team to that location for a closer look. If the location in question is difficult to access on-foot or contains dangerous terrain features, this decision adds significant safety risks to ground personnel, as well as risks to the search mission itself (see Fig. 1.1).

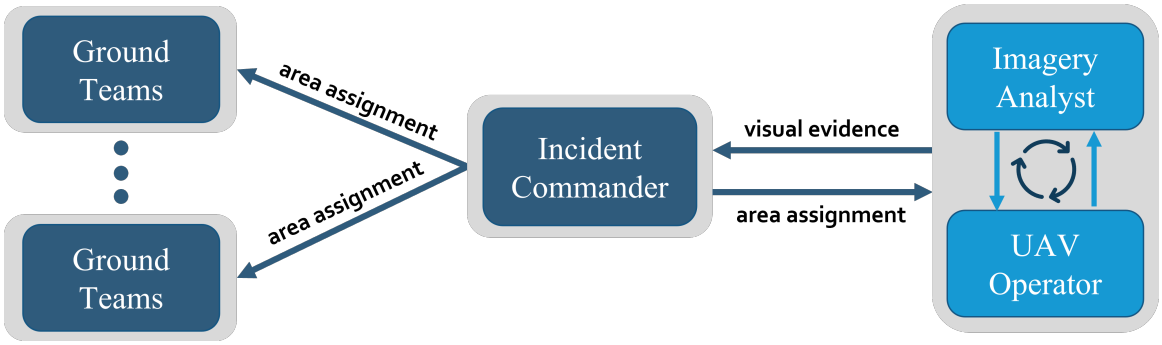


Figure 1.2: WiSAR Concept of Operations: search teams receive area assignments from the Incident Commander. Once assigned, the UAV Operator executes the flight path while the Imagery Analyst processes the video stream to provide visual evidence of the missing person's location. The UAV's flight path is often modified in response to the need to mitigate occlusion in order to enable high-confidence imagery analysis.

On the other hand, if the imagery supports a negative detection with high-confidence, the Incident Commander can direct the ground team to higher-probability areas, thus improving the efficiency of the search. Therefore, the way in which imagery is collected is of vital importance, and this process is driven by the flight path executed during the mission, be it manually executed by the pilot or semi-autonomously controlled by an algorithm.

1.2 Challenges for UAV-Enabled WiSAR

While many commercially-available UAVs are equipped with high-resolution imaging sensors and some form of semi-autonomous mission planning software, the majority of UAV operators choose to fly manually when supporting WiSAR missions, especially during the initial phase of the operation [5]. The primary reasons for this are (a) wilderness terrain is complex and can be challenging for automated flight planners, and (b) there is often a need to continually reposition the UAV when the human operator or imagery analyst detects some feature or clue in the video stream worthy of further investigation [5, 6].

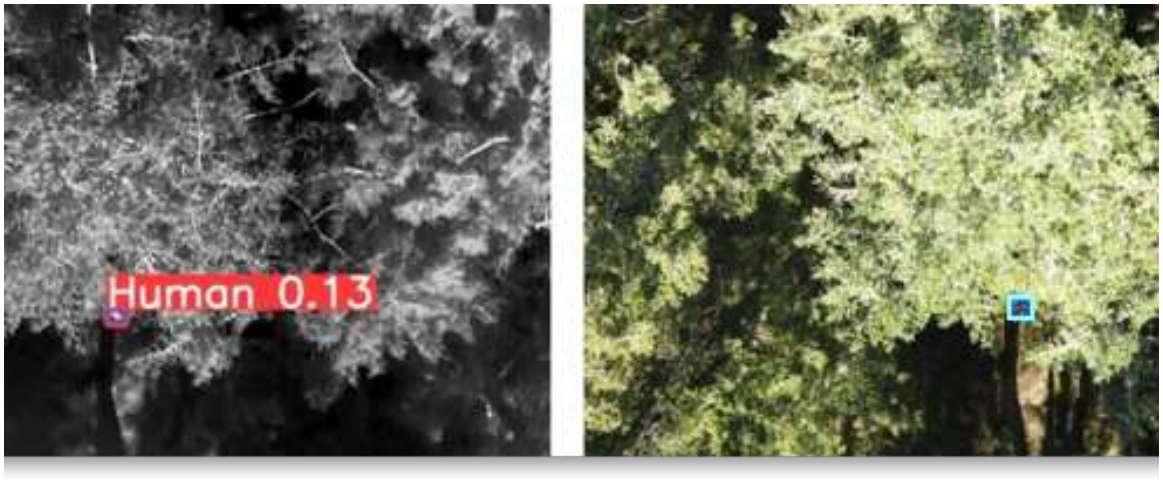


Figure 1.3: Example of occluded view affecting object detection in thermal (left) and visual (right) images of a wilderness scene. The object detector outputs a low-confidence detection (orange box) near the ground truth location for the human (blue box) in the thermal image, but fails to detect the human in the visual image.

Even in situations where pre-planned waypoint missions are used, operators and imagery analysts face many additional challenges, including:

1. Occlusion from dense vegetation and terrain features
2. Short operational flight times due to limited battery life
3. Video and control data link interference
4. Regulatory restrictions for beyond visual line-of-sight operations
5. Small object detection from aerial perspective scenes

Of these challenges, occlusion is the most difficult to deal with from a technological standpoint. Detecting objects in the presence of partial occlusion is difficult for trained human imagery analysts, and computer vision-trained object detectors tend to perform poorly in occluded conditions as well. Fig. 1.3 illustrates the difficulty of identifying humans from overhead imagery when the subject is partially occluded. The images shown in this figure are representative of those collected during actual WiSAR missions, and the object detector was trained on a purpose-built dataset for

detecting humans from aerial imagery in wilderness environments [7]. This motivates the need for coverage planners that account for and mitigate the effects of occlusion for the detection task.

1.3 Research Objectives, Contributions, and Assumptions

Existing coverage path planners are insufficient for addressing the challenges of UAV-enabled WiSAR previously described. More specifically, we propose that an ideal coverage planner for WiSAR should have the following characteristics:

1. **Completeness:** Achieve complete coverage of the search area
2. **Occlusion-Awareness:** Generate viewpoints which consider the geometry of objects that cause occluded views
3. **Energy Efficiency:** Produce paths which maximize usable flight time

Developing this capability requires further research in order to answer several key questions. First, in what sense is the coverage considered complete, and what metrics can we employ to measure search coverage and quality? How can we detect occlusion and, once detected, generate viewpoints which see around the occlusion? Lastly, how do the objectives of completeness, occlusion-awareness, and efficiency interact, and what are the tradeoffs between these objectives? Addressing these questions is the primary focus of this research.

Contributions

The research presented in this dissertation investigated the occlusion-aware coverage planning problem for UAVs, with particular emphasis on the WiSAR application. The primary contribution of this research is an offline occlusion-aware coverage mission planner for UAV-aided WiSAR; a capability that is not currently available with off-the-shelf UAV mission planners. The proposed system operates on a 3D geometric environment model derived from LiDAR survey data, offers guaranteed coverage

with respect to that model, obeys geofence and altitude constraints, and generates battery-efficient plans that are compatible with existing UAV software applications. Additional research contributions include:

- An altitude-constrained aerial viewpoint generation method which approximately solves the UAV variant of the 3D Art Gallery Problem (AGP)
- An image-based processing pipeline for detecting individual tree locations from LiDAR point cloud data and constructing a geometric proxy of the environment
- A simulated performance analysis of the proposed occlusion-aware planner
- A real-world demonstration of the proposed occlusion-aware planner as applied to a search and rescue scenario

Assumptions

For this research, it is assumed the UAV used for WiSAR is a small (backpackable) multirotor, equipped with visual and thermal cameras and the ability to transmit high-definition video and telemetry over a wireless datalink at distances up to 5km. The UAV is assumed to be capable of a minimum flight time of 25 minutes (conservative estimate with full computational loading). For simplicity, the cameras are body-fixed in a nadir orientation (pointing straight down); future work will consider the gimbaled camera scenario. These assumptions are based on the actual hardware and software specifications of the UAV used for this research and are a realistic representation of similar systems used in the field.

The planner is assumed to have access to previously-obtained LiDAR survey data or 3D point cloud data of the search area, which may be outdated but is representative of the underlying terrain and (to a lesser extent) the vegetation in the environment. While high-resolution point cloud data is available in many areas in the U.S., it is not available globally. However, it may be possible to extend the methods presented to lower-resolution elevation and vegetation density estimates from satellite-based

sensors which do provide global coverage (e.g., NASA's Shuttle Radar Topography Mission [8]). Lastly, the human subject of the search effort is assumed to be stationary.

1.3.1 Organization

The rest of this dissertation is organized, as follows. Chapter 2 covers related work and relevant background for understanding the coverage planning methods, which includes separate discussions on the problems of viewpoint generation and path planning, as well as a description of techniques for identifying trees from LiDAR. Chapter 3 defines the theory and methods for altitude-constrained occlusion-aware viewpoint generation used in the proposed solution. Chapter 4 describes the complete altitude-constrained occlusion-aware coverage planner, and includes simulated results and statistical performance analysis. The results and analysis from real-world experiments are documented in Ch. 5, and Ch. 6 provides conclusions and outlines several ideas for future work.

Chapter 2
BACKGROUND AND RELATED WORK

This research focuses on the occlusion-aware coverage path planning problem as applied to using small, computer vision-enabled UAVs for search and rescue operations in wilderness environments. The challenging aspects of this research involve dealing with two classical NP-hard computational geometry problems: the 3D Art Gallery Problem (AGP) of finding a minimum set of viewpoints that guarantee complete area coverage for a given sensor footprint, and the Traveling Salesman Problem (TSP) of finding an optimal path connecting those viewpoints. To address the inherent complexities associated with these two problems, this research builds upon techniques developed for structural inspection planning and mapping, graph theory, machine learning, optimization, and Satisfiability Modulo Theory (SMT). The relevant concepts from these topics are provided in what follows.

2.1 Classical UAV Coverage Planning Methods

The traditional framing of the coverage path planning problem for aerial search is to find feasible trajectories in 3D space that enable complete sensor coverage of an area-of-interest while satisfying constraints related to vehicle dynamics, sensor limitations, and environmental hazards [9]. This topic has been extensively studied in the literature [10–13], and [14] provides a recent survey of existing methods. Many approaches to the coverage planning problem rely on decomposing the environment into grids, graphs, or polygons as a function of a sensor footprint model, and then connecting these vertices or regions together to form a path that is, in some sense, optimal, where optimality is defined in terms of minimizing time-to-detect or time-to-

complete, minimizing fuel or control effort, or minimizing the total path length [15]. By construction, a path which visits each node is guaranteed to provide complete sensor coverage of the area-of-interest. The key assumption for methods of this kind, referred to in the literature as cellular decomposition or boustrophedon decomposition, is that all desired information contained in a region or node can be recovered from a single viewpoint. This assumption is valid for applications that don't require unoccluded views of the ground (for example, creating orthomosaics to support forest growth analysis). For the WiSAR task, however, making such an assumption would not provide adequate coverage to enable the detection of humans, who may be obscured by terrain features or vegetation, from aerial imagery.

While classical decomposition methods are insufficient for WiSAR, modern coverage planning methods specifically designed for structural inspection and mapping applications do have utility for search applications. Despite the difference in objectives (high-accuracy 3D reconstruction versus high-confidence unoccluded viewpoint coverage), the inspection and search applications share common challenges in viewpoint generation and path planning. The next two sections provide an overview of coverage planning methods designed for 3D scene reconstruction and mapping.

2.2 Coverage Planning for Inspection and Mapping

The literature around structural inspection planning and mapping for UAVs provides many insights into occlusion-handling that can be applied to the aerial search application. In particular, this body of literature frames the coverage problem in a way which explicitly considers viewpoint quality. Specifically, the problem is partitioned into two sub-problems: viewpoint generation, which can be thought of as an AGP, and path planning, which can be construed as a TSP. The AGP and TSP are long-standing NP-hard problems, and approximation methods exist [16–21], the advantages and limitations of which are discussed in more detail in Sections 2.3.1 for the AGP, and 2.4 for the TSP.

Maboudi, et al., [22] provide a recent survey of approaches for viewpoint generation and path planning for 3D reconstruction and inspection applications. Existing approaches are categorized as model-based or model-free, depending on whether the planner has access to a pre-existing model, or geometric proxy, of the object or structure-of-interest.

2.2.1 Model-Free Coverage Planning

Model-free viewpoint planning methods iteratively generate and update an object model of an object or structure as new measurements are taken, thus these methods are online and must employ an exploratory strategy. Categorically, these approaches are either frontier-based, in which case the path evolves according to a defined boundary between explored and unexplored space, volumetric-based, in which the 3D spatial information is included in addition to the frontier for viewpoint evaluation, or surface-based, when information about the shape and trend of the object surface (often represented by a mesh) is used to derive the next viewpoint. Most model-free methods use the next-best-view strategy [23], which is a greedy algorithm that determines the best viewpoint based on an information heuristic derived from the current state of the output model (e.g., the area with the largest unexplored volume or area [24]). These approaches are, in many cases, combined with a simultaneous localization and mapping (SLAM) module, which is used to direct and estimate the state of the UAV while reducing the impact of measurement uncertainty on the reconstructed map or model. However, even though model-free approaches offer fast exploration, they inherently cannot guarantee complete coverage of an object due to the use of local search strategies and non-linear objective functions, a combination that is susceptible to local minima [22].

2.2.2 Model-Based Coverage Planning

Model-based methods generate viewpoints of a structure by leveraging an existing coarse representation of the target structure, known as a geometric proxy, typically represented in the form of a mesh surface, point cloud, or voxel occupancy map. This approach is almost always accomplished in two phases, where the goal of the first phase is to explore the structure to generate the geometric proxy and establish safe airspace boundaries, followed by a second phase, which uses the collected knowledge of the scene geometry to facilitate a globally-optimized viewpoint set, where the definition of optimality varies by application, but is often a measure of the accuracy of the 3D reconstruction. This is known as the explore-then-exploit strategy and is the basis for the approach proposed in this dissertation.

The majority of model-based strategies generate a geometric proxy via an exploratory flight, the goal of which is to collect multi-view or overlapping nadir imagery, or point cloud data. These data may have holes/gaps and inaccuracies due to the fact that exhaustive coverage with multiple passes from various aspects is not typically not practical, but in general are usually sufficient for low-precision 3D reconstruction using well-established algorithms for this task, such as structure from motion (SFM) or multi-view stereo (MVS). In most cases, initial data collection flights can be automated with off-the-shelf planners, usually limited to simple patterns (e.g., circular orbit, lawnmower) at a safe attitude above the structure. In more difficult scenarios with complex geometries, these conservative patterns may not generate sufficient samples for reconstructing a geometric proxy, which necessitates the need for manual piloting [22].

Some proposed model-based methods forego the exploratory flight in favor of a pre-existing data sources, including Google satellite imagery, street and building maps, resulting in usable 3D surface or building models derived from photogrammetry [25, 26]. While less available and significantly more expensive to collect than

their image-based counterparts, laser-scanning based sources (i.e., LiDAR surveys) are more accurate and have also been used successfully for generating geometric proxies in 3D reconstruction algorithms [22, 27].

2.3 Viewpoint Generation and the Art Gallery Problem (AGP)

The exploitation phase of model-based coverage planning consists of generating candidate poses and then computing the minimum set of poses needed for optimal coverage. Some strategies for generating candidate poses include sampling the geometric representation [28], or assigning fronto-parallel views to facets of the underlying structure (i.e., for each viewpoint, the camera's line of sight is effectively perpendicular to a mesh surface component) [29].

Bircher, et al., [28] propose a sampling-based, probabilistically complete, 3D path planner specifically designed for structural inspection. While not intended for autonomous search, the authors apply the algorithm to a mountain landscape environment with a search and rescue mission scenario as the inspiration. The algorithm requires an existing 3D model of the environment, which is down-sampled into a 3D mesh. This mesh is used to check for occlusion and collision. The path produced also considers vehicle dynamics and sensor characteristics, such as field of view and resolution, and can constrain the path to ensure the incidence angle from the UAV to the landscape does not exceed a maximum threshold. The planner is based on viewpoint sampling and Rapidly-Exploring Random Trees (RRT*) [30], and relies on solving several convex subproblems in a multi-optimization scheme.

Another viewpoint generation method is to construct a volumetric occupancy map where each voxel is assigned an observation quality score and is classified as occupied, free-space, or unobserved. These methods often involve visibility testing, which involves a computationally expensive ray casting procedure to evaluate which voxels are visible from each viewpoint, thus most methods perform this computation off-line (with a few exceptions) [22].

2.3.1 The Art Gallery Problem

The problem of finding the minimum set of coverage viewpoints can also be thought of as an Art Gallery Problem (AGP). The AGP is a well-studied problem in computational geometry and consists of finding the minimum number and placement of guards sufficient to provide complete coverage of an art gallery represented by an arbitrary polygon. Solving this problem is, at a minimum, NP-hard, however recent work has discovered the art gallery problem belongs to the $\exists\mathbb{R}$[1] complexity class, and is, in fact, $\exists\mathbb{R}$-complete. The $\exists\mathbb{R}$ class, which is sometimes referred to as the "real analog" of NP, is used to describe the computational complexity of decision problems with real-valued parameters [31]. Many algorithms for approximating the solution to this problem exist for the 2D case, most of which involve decomposing polygons into convex parts and placing a guard in each convex region [19]. For a polygon with n vertices, this procedure can be computed in $O(n^4)$ and yields an approximation that is, at most, $O(n \log n)$ times the optimal [19]. However, this type of decomposition is not practical for 3D polyhedra and can result in a representation with an intractable number of components [32]. Furthermore, the concept of a visibility graph, which is a fundamental representation useful for 2D AGP approximations, is not easily extended to 3D space [33, 34]. Interestingly, a stochastic gradient descent formulation of the AGP has only recently been investigated in a thesis published in 2022, however performance of the algorithm proposed in this work is sensitive to polygon shape, initial guard placement, and other hyperparameters, and thus is not scalable in its current form [35].

Approximations for the 3D AGP typically require making simplifying assumptions about the surface structure. For example, Marzal [20] developed an approximation technique for the special case of positioning vertex guards for covering orthogonal pseudo-polyhedra; a useful result for urban environments where occlusion from build-

[1] This complexity class is pronounced "existential-over-reals," or "E-R" (as in: Emergency Room), for short. It can be shown that NP $\subseteq \exists\mathbb{R}$ [31].

ings can be modeled as orthogonal polyhedra. A similar assumption about the underlying surface was made to simplify visibility testing in [21], which would also be useful for urban environments, but would fail in wilderness environments due to the complexity of the terrain.

Another approach considers a more generalized class of 3D polyhedra and leverages insights from the 2D AGP (namely, triangulation and three-coloring) with an added heuristic for adjusting the z-value of the pose to ensure complete coverage for a UAV with a specified sensor field of view [36]. This is known as the Vantage Waypoint Set Generation Algorithm (VWSGA). A modified version of this method was applied to the coverage path planning problem for UAVs in [37], which was proven to work well in simulated environments where terrain occlusion is represented by a particular class of polyhedra. The research presented in this dissertation builds on the VWSGA, so a complete description is provided when discussing methods in Chapter 3.

Viewpoint Generation with Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs) provide a scalable means for performing inference on graphs with arbitrarily complex data types, a task which has traditionally proven challenging for machine learning algorithms. One increasingly popular application of GNNs is for solving NP-hard combinatorial optimization problems, including the TSP, SAT, knapsack, minimum vertex cover, and maximum cut [38]. Curiously, the application of GNNs specifically to the 3D AGP has not yet been attempted to the best of our knowledge [38, 39]. Due to this novelty and applicability to viewpoint generation, a short investigation was conducted into the application of GNNs to the 3D AGP, and initial results are promising. While not central to the theme of this dissertation, a full description of this work and initial results are provided in Appendix A for the interested reader.

2.4 Path Planning and the Traveling Salesman Problem (TSP)

Once a set of covering viewpoints is defined, the next step is to compute a feasible or optimal path connecting the viewpoints, where optimality is usually defined in terms of minimum cost (e.g., the shortest distance traveled or shortest time) to complete the route. This is known as the Traveling Salesman Problem (TSP), which is a fundamental combinatorial optimization problem that is proven to be NP-hard to solve [40]. The TSP can be posed as follows: given a graph $G(V, E)$, where V is the set of vertices and E is the set of weighted edges, determine if there exists a Hamiltonian cycle (a tour which visits each vertex exactly once) which minimizes the sum of edge weights [41]. A slightly relaxed variant of this problem is the Metric TSP, where the minimum cost tour visits each vertex *at least once*, and metric implies that the edge weights are non-negative, symmetric, and satisfy the triangle inequality [40].

Methods for computing a solution to the TSP and its variants are either exact or heuristic [42]. Exact methods include brute force, dynamic programming, and branch and bound solvers. These solvers find globally optimal Hamiltonian cycles, but their time requirements grow exponentially with the graph size and therefore can only be used on "small" graphs, where small is a function of compute and memory resources. A good rule of thumb for exact methods (excluding brute force[2]) is 15 vertices or fewer. Thus, larger TSPs must be approximated with heuristic solvers. While these methods are subject to local minima, they do surprisingly well in practice [42].

2.4.1 Heuristic TSP Solvers

All heuristic TSP approaches employ a neighborhood search and perturbation scheme. Comparisons between the cost of a candidate solution are compared with the cost of a permuted version of that sequence. One of the simplest and efficient schemes for

[2]For a 15 node TSP, a computer which can check one million permutations per second would take roughly 41 years to evaluate all 15! permutations, whereas this problem may be solvable with dynamic programming or branch and bound methods in minutes.

local TSP search are 2-opt and 3-opt, which operate by iteratively swapping edges in a candidate tour until no improvement in cost can be achieved [43]. The Christofides-Serdyukov [44] is the one of the best-known TSP algorithms and offers a performance guarantee that the returned solution will be within a factor of 3/2 of the optimal solution. This algorithm is based on finding a minimum spanning tree for G and then adding the minimum cost matching on the odd degree vertices. While this approach has stood the test of time for nearly four decades, recent research yielded a (slightly) improved approximation using a similar approach with random spanning trees [40]. The Lin-Kernighan heuristic for TSP is more complex heuristic which has become a popular choice due to its ability to find *optimal* solutions for large-scale TSPs (e.g., 13,509 vertices) at a high frequency despite being an approximate algorithm [16, 28]. Another common option for heuristic search is known as simulated annealing, which is a metaheurstic that includes an element of random exploration, which allows the algorithm to escape local minima.

2.4.2 The Satisfiability Modulo Theory (SMT) Approach to the TSP

Satisfiability Modulo Theory (SMT) is the problem of determining whether a mathematical formula is satisfiable or not. It is an extension of the Boolean satisfiability problem (SAT) but is much more expressive in that it can evaluate the satisfiability of complex formulae involving real numbers, integers, and various data structures. Recent work modeled the coverage planning problem for the multi-agent case (commonly known as the Vehicle Routing Problem) as a SAT problem [45, 46]. In this work, the coverage planning problem for an aerial survey with coverage overlap requirements is encoded as at SAT problem using one-hot encoding[3] with SMT-specified constraints. A sequential SMT solver is employed, which iteratively solves the optimization problem and improves the solution with each iteration. This approach

[3]One-hot encoding refers to a technique used in machine learning for representing categorical data via indexing into binary vectors.

offers distinct advantages over other TSP solvers in that it is faster than existing optimization approaches, and is "anytime feasible," meaning the algorithm can be halted anytime after the first iteration and the solution provided is guaranteed to be feasible. Furthermore, this method explicitly considers the launch and recovery point for the UAV and minimizes the inefficiencies due to backtracking through a previously covered area. An open-source tool developed in [46], which is known as the WADL Coverage Path Planner, exists and is among the state-of-the-art for solving the multi-agent TSP in large area surveys.

While the flight plans generated with WADL are not specifically designed to be occlusion-aware, this method could be adapted to solve the occlusion-aware coverage problem for the multi-agent case. Doing this would require replacing the input graph, which is a grid graph encoding the field-of-view overlap requirements, with an "occlusion-aware" graph, which is a concept developed in this dissertation. Furthermore, the current WADL implementation assumes a constant survey altitude, so some adaptation is needed to account for a variable altitude, which is a significant factor when considering field-of-view and occlusion constraints. Appendix B provides such a formulation for the interest reader.

2.5 Individual Tree Detection and Segmentation (ITDS)

The occlusion-aware planner proposed in this dissertation relies on the ability to generate a geometric proxy of an area which captures occlusion information needed for occlusion-aware waypoint planning from pre-existing LiDAR point cloud data. In a wilderness setting, one crucial aspect of this problem is the ability to detect, spatially locate, and determine the extents of trees from LiDAR data. In the remote sensing and digital signal processing literature, the problem of locating trees is a target recognition problem known as individual tree detection (IDT), whereas individual tree segmentation (ITS) deals with the classification problem of associating points in a point cloud to individual trees; the combination of these two tasks is also re-

ferred to as individual tree detection and segmentation (ITDS) [47, 48]. Approaches to ITDS are can either be point cloud-based or raster-based, in which case images or LiDAR-derivative raster products, such as digital surface model (DSMs), digital terrain models (DTMs), or canopy height models (CHMs) are processed [49, 50]. A DSM is an image (or grid) created by the point-to-raster method, in which each grid cell is assigned the highest elevation return from the point cloud at that point. A CHM is a DSM in which the influence of the underlying terrain has been removed, leaving only the above ground measurements. A DTM is a gridded representation of only the ground-level elevation and is used for normalizing point clouds and creating CHMs [51, 52].

Raster-based strategies aim to find local maxima within the CHM in order to spatially locate trees, and then apply a segmentation scheme to define individual tree crowns [47]. Methods of this kind are effective in forests with well-defined crowns and some amount of spatial separation between neighboring trees. Crowded forests with overlapping canopies obscure tree locations in the IDT step and blur the boundaries between individual trees, ultimately leading to bias and inaccuracies in the segmentation step. Additionally, raster-based methods often require setting numerous parameters and thresholds which must be tailored to individual scenes, and often require some degree of human intervention and expertise to yield consistently accurate results. Point cloud-based ITDS methods usually employ clustering algorithms and are typically more accurate but require more computation compared to raster-based approaches. One of the primary challenges which limits the accuracy of both strategies is broad-leaved forests with overlapping or irregular canopies [50].

2.6 Ground Control Station (GCS) and Mission Planning Software

Many commercially-available ground control station (GCS) software solutions exist which offer some form of automated survey or mapping, including QGroundControl, Pix4D, DJIPro, Mission Planner, DroneDeploy, DroneHarmony, and UgCS [53–59].

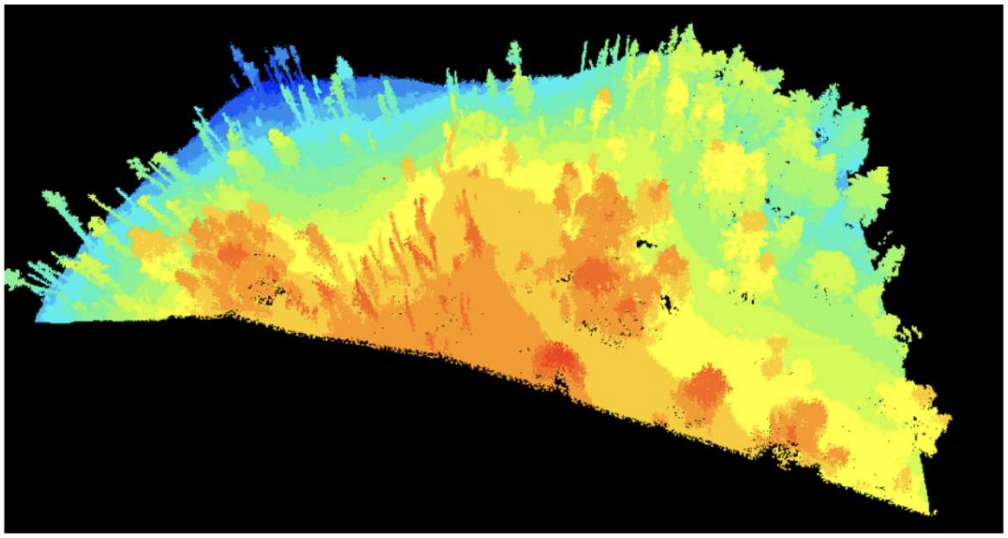


Figure 2.1: LiDAR point cloud of Chelan County, Washington, area from a survey conducted in 2017.

Many of these options are paywalled or subscription-based, but include high-end data processing which can be used to generate elevation models or 3D reconstructions from the images or videos collected during the acquisition phase. The path generation capabilities of these off-the-shelf planners offer limited settings and generally produce variants of a lawnmower or cross-grid pattern, with some additional features (e.g., terrain-following). As of this writing, off-the-shelf GCS applications do not offer an autonomous, occlusion-aware search mission path generation capability; the research proposed in this dissertation aims to develop this capability in a way that is compatible for use with existing GCS applications.

Chapter 3

OCCLUSION-AWARE VIEWPOINT GENERATION FROM GEOMETRIC PROXIES

This chapter describes the theory and methods used to generate aerial viewpoints for occluded environments. It begins with an in-depth overview of the Vantage Waypoint Set Generation Algorithm (VWSGA) [36] developed by Savkin and Huang. An adaptation of the VWSGA which aims to address the practical limitations of the original algorithm is then described in detail; this new algorithm is referred to as the VWSGA+. Lastly, the raster-based methods used for generating geometric proxies of wilderness environments from LiDAR point clouds are provided.

3.1 The Vantage Waypoint Set Generation Algorithm (VWSGA)

The Vantage Waypoint Set Generation Algorithm (VWSGA) [36] aims to approximately solve the drone version of the 3D Art Gallery Problem (AGP) for finding aerial viewpoints which completely cover environments comprised of complex and uneven terrain. This algorithm offers mathematical coverage guarantees while avoiding many of the computational complexities of the AGP by employing a two-stage approach. In the first stage, the problem is converted into a graph, and the 2D coordinates for each point in the vantage point set are determined by applying a series of transformations on this graph. In the second step, the altitudes of the vantage set are computed from geometric first principles, while also accounting for user-defined constraints, such as field of view and minimum allowable altitude.

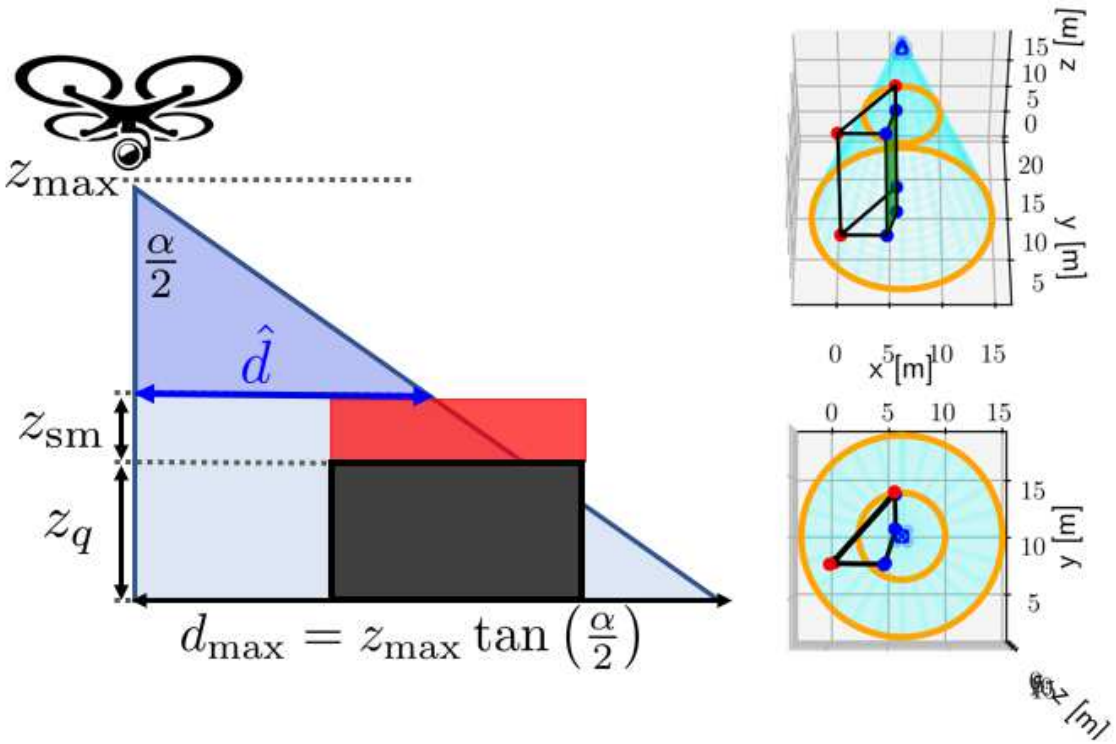


Figure 3.1: Conic FOV Coverage Model. (Left) 2D FOV geometry with triangulation side length constraint ($\hat{d}$) highlighted in blue. (Right) 3D FOV geometry for $\alpha = 30°$, green face and blue vertices are visible, red are either outside the FOV or not in direct line-of-sight to the UAV. Level sets of the visibility cone are shown in orange at ground-level and at the height of the polyhedron.

3.1.1 VWSGA Terrain and Coverage Models

The terrain model considered by the VWSGA is illustrated in Fig. 3.2. The model consists of flat areas and non-flat areas. The non-flat areas are modeled by polyhedra formed by two polygons, a top face and a bottom face, each with the same number of vertices, which results in quadrilateral side faces. The top and bottom polygons and resulting polyhedron need not be convex, however the top vertices are constrained such that their projection onto the xy plane lie inside the bottom polygon (this geometric object is referred to as a *truncated pyramidal frustum*). The top face polygons are not required to be parallel with the bottom polygons, but the vertices must lie on the same plane.

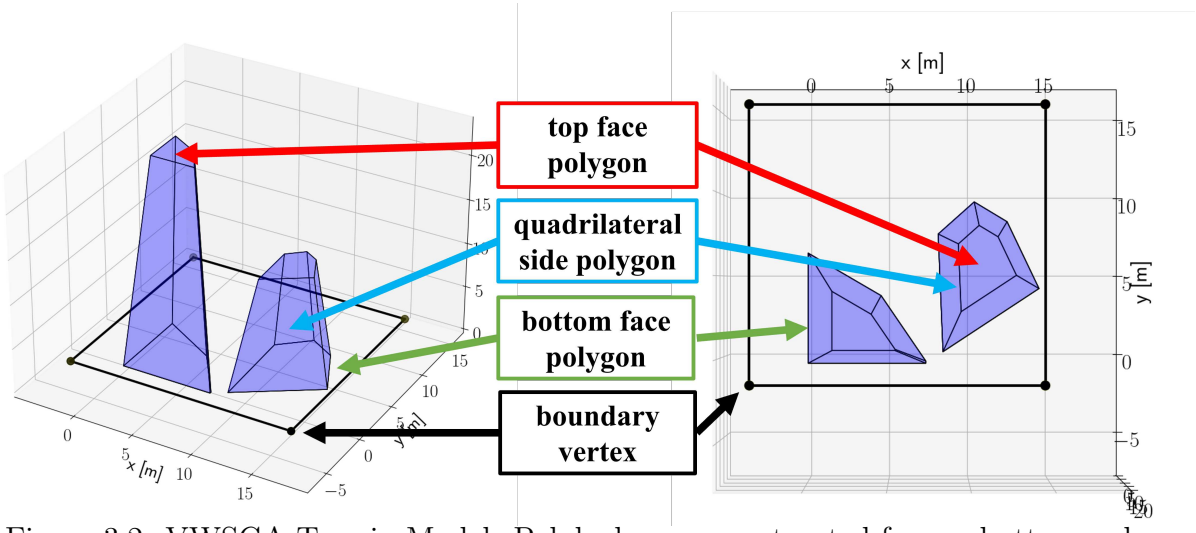


Figure 3.2: VWSGA Terrain Model. Polyhedra are constructed from a bottom polygon and top polygon with the same number of vertices. The $z = 0$ projection of top polygon vertices must lie within the bottom polygon area, thus creating side quadrilaterals that are canted.

The VWSGA's model allows slight variations in the flat areas of the terrain, which are accounted for by the variable ε, which is a small, non-negative value representing the maximum difference between any two points in the flat regions of the map.

Given this geometric proxy, the VWSGA computes a set of vantage points which guarantees, by construction, all top and side faces of the polyhedra, as well as all points within the flat areas, will be visible to at least one vantage point in the set. In this context, a point on the terrain or polyhedron is visible if two conditions are met. First, the point must lie within the field-of-view cone, which can tested by checking if the point-in-question is within the radii defined by $z \tan(\alpha/2)$, where z is the height of the vantage point. Second, the line segment between the vantage point and point on the terrain must not intersect any polyhedron, i.e. there must be a clear line-of-sight from the vantage point to the point. Figure 3.1 illustrates the relevant geometries related to coverage. The orange circles represent the coverage cone radii defined at different values of z, one for the ground plane ($z = 0$) and one corresponding to the height of the polyhedron's top face vertices.

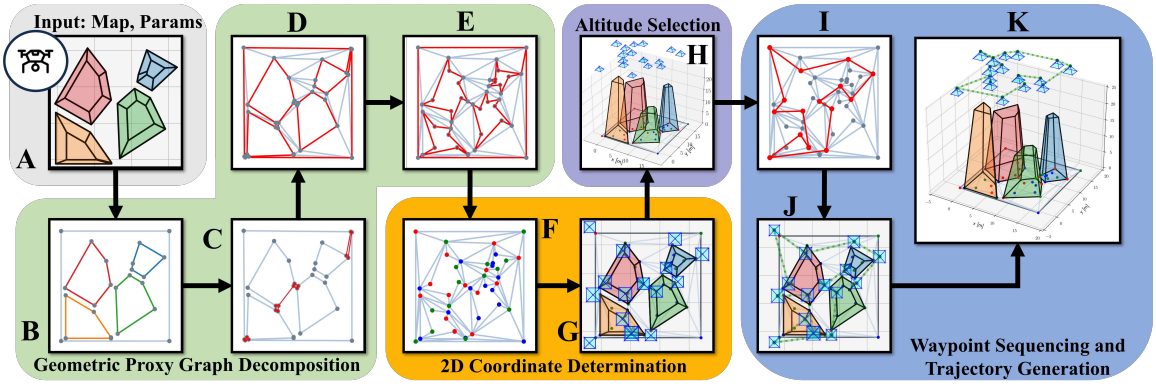


Figure 3.3: VWSGA and Waypoint Sequence Generation Diagram. Input polyhedron map and mission parameters (A); initial geometric proxy graph with boundary and bottom polygons only (B); graph after hole removal and polygon merge procedure (C); triangulated graph (D); extended triangulation which includes top polygon vertices (E); three-colored graph (F) and 2D waypoint coordinates (G); 3D waypoints from altitude selection (H); TSP solver solution (I) and waypoint sequence (J and K).

3.1.2 VWSGA Graph Decomposition and Processing

Figure 3.3 depicts the core elements of the VWSGA process, which include Geometric Proxy Graph Decomposition, 2D Coordinate Determination, and Altitude Selection. The input to the algorithm is a geometric proxy (or polyhedron map) of the environment, and mission parameters, such as a minimum allowable vantage point altitude ($z_{\min}$), a vertical safety margin distance between the vantage point and terrain (z_{sm}), and a conic field-of-view angle (α).

From these inputs, a graph representation of the environment is created, and several transformations are applied in what will be referred to as the Geometric Proxy Graph Decomposition module (Fig. 3.3, steps B-E). The goal of this module is to create a graph that is guaranteed to have a three-coloring exist, which is essential for the 2D waypoint determination. Initially, the graph is constructed using only the vertices and edges from the bottom polygons and map boundary (Fig. 3.3 B). This graph representation can be thought of as a polygon with holes.

Algorithm 1 Algorithm for removing holes in a simple polygon

```
function CLOSESTPAIR(vertexSetA, vertexSetB)
    minDist ← ∞
    minDistPair ← (null, null)
    for v_a in vertexSetA do
        for v_b in vertexSetB do
            dist ← norm(v_b-v_a)
            if dist<minDist then:
                minDistPair ← (v_a, v_b)
                minDist ← dist
            end if
        end for
    end for
    return minDistPair
end function
function CUT AND MERGE POLYS(M)
    for polygon in M do
        vertexSetA ← vertices in polygon
        vertexSetB ← M - vertexSetA
        edge ← closestPair(vertexSetA, vertexSetB)
        M ← add edge(edge)
        M ← clone edge and rewire(M, edge, offset factor)
    end for
    return M
end function
```

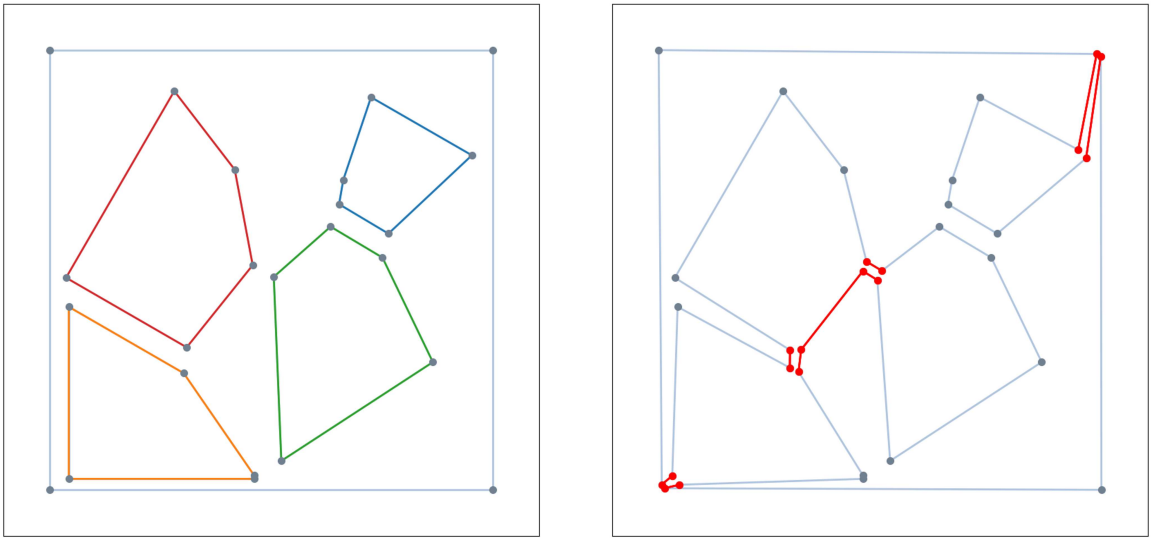


Figure 3.4: Graph before (left) and after (right) hole removal and polygon merge procedure, with exaggerated spatial offset for clarity. The red highlighted edges and vertices were used in the merge procedure, and the resulting edges of the graph trace out a simple, non-intersecting polygon.

Next, these holes are effectively removed through a polygon cutting and merging procedure. This procedure (Fig. 3.3 C) involves connecting the holes together with non-intersecting lines, and then connecting one of the holes to the boundary. Each of these lines and their associated endpoint vertices are duplicated (with some spatial offset) and the edges of the polygons are rewired such that the end result forms a simple, non-intersecting, hole-less polygon. An illustration of this process is shown in Fig. 3.4, where the spatial offset has been exaggerated to visually accentuate that the newly added edges form the boundary of a single, non-intersecting polygon. Algorithm 1 provides pseudocode for the cut and merge procedure given an input map of bottom face polygon vertices, $\mathcal{M}$. The *closestPair* function takes as input two sets of vertices and returns the pair of vertices corresponding to the minimum Euclidean distance (with respect to all vertex pairings between the two input sets).

The hole-less polygon is then triangulated using Constrained Delaunay Triangu-

lation, in which the triangulation must include the edges of the polygon as triangle sides in the result [60]. This is shown in Fig. 3.3 D, in which the red edges were used as constraints in the triangulation. Next, the triangulation is extended by incorporating the top vertices and associating each one with a bottom polygon edge to form a triangle. The output of this process is a triangulated simple polygon which includes all vertices from the original polyhedron map.

3.1.3 2D Coordinate Determination

Next, the geometric proxy graph is three-colored such that no two vertices connected by an edge in the triangulation share the same color. This groups the vertices into three distinct sets. In general, determining whether a three-coloring exists for an arbitrary graph is an NP-complete problem [61]. However, such a three coloring is guaranteed to exist because any simple triangulated polygon (i.e., the output of the Geometric Proxy Graph Decomposition module) can be three-colored [19]. Furthermore, once the first triangle is colored, the remaining vertex colors are determined, so from an algorithmic perspective, the three-coloring can be executed with a single pass through the set of vertices.

An implementation of the Koosheh and Moret algorithm for three-coloring a triangulated simple polygon is used for this research [62]. This algorithm is a simple $O(n)$ method for three-coloring which only requires a perimeter-ordered set of vertices $p_0, \cdot, c_n$ for a given triangulated simple polygon $\mathcal{T}$. Given this perimeter-ordered list of n vertices, and colors labeled with the number 1,2, and 3, the algorithm is:

It can be seen how a perimeter-ordered set of vertices is obtained during the Geometric Proxy Graph Decomposition stage in Fig. 3.3. Specifically, an initial polygon perimeter is established during the hole removal and polygon merge procedure, which is shown by the highlighted red edges in (D) in Fig. 3.3. This perimeter is then adjusted during the extended triangulation phase to include the top polygon face vertices, as shown in (E) in Fig. 3.3. Tracking the perimeter throughout the con-

Algorithm 2 Kooshesh and Moret's Three-Coloring Algorithm [62]

```
Color(p_0) ← 1
Color(p_1) ← 2
for i=1 to n-1 do do
    if odd(deg(p_i)) then
        Color(p_{i+1}) ← Color(p_{i-1})
    else
        Color(p_{i+1}) ← 6 - Color(p_{i-1}) - Color(p_i)
    end if
end for
```

struction of the graph avoids the need for obtaining the perimeter using a convex hull or other algorithmic approach. Once three-colored, the subset of vertices belonging to the color with the minimum number of elements forms the basis for the vantage point set and defines the x and y values for each vantage point. Furthermore, any duplicate vertices from the polygon cutting and merging procedure which share the same color can be considered to be a single vertex, since the duplication is artificial and both vertices represent the same physical location. Once the 2D coordinates are established, the z value of each of these vantage points is assigned based on several factors, as discussed in the following section.

3.1.4 Altitude Selection

Given the 2D coordinates of the vantage waypoint set, the VWSGA selects altitudes which ensure complete coverage of all polyhedron faces and all points in the flat areas of the map. Define $\mathcal{M} = \{v_i \in \mathbb{R}^3 | i = 0, \ldots, n-1\}$ as the set of all n vertices in the geometric proxy. Let $\mathcal{B} \subset \mathcal{M}$ denote the subset of vertices belonging to the boundary of the map, and let $\mathcal{P} \subset \mathcal{M}$ represent the subset of vertices which belong to the polyhedron. Define $\mathcal{T} = \{(v_a, v_b, v_c)_j | v \in \mathcal{M}, j = 0, \ldots, k-1\}$ denote the set

of vertex tuples which define the triangles resulting from the triangulation procedure. From the vantage point set, chose one vertex belonging to each polyhedron that will be assigned the responsibility of viewing the entire top face of that polyhedron; this set is denoted as $\mathcal{P}_\star \subset \mathcal{P}$. Furthermore, for any vertex $v_i \in \mathcal{P}$, let $\mathcal{Q}_i$ be the set of vertices which form the two side quadrilaterals with v_i, and $\mathcal{F}_i$ be the set of vertices which form the top face of the polyhedron for which v_i is a vertex. Then, the relevant parameters for selecting the vantage point altitude for a given vertex v_i are:

- $d^i_{\mathcal{T}}$: maximum triangle side length associated with v_i
- d^i_q: maximum distance to vertices in $\mathcal{Q}_i$
- z_q: maximum z-value of the vertices in $\mathcal{Q}_i$
- d^i_f: maximum distance to vertices in $\mathcal{F}_i$
- z_f: maximum z-value of all vertices in $\mathcal{F}_i$

Vantage point altitudes are then assigned according to the following equation.

$$z_i = \begin{cases} \max(\epsilon + \frac{d^i_{\mathcal{T}}}{\tan\left(\frac{\alpha}{2}\right)}, z_{\min}) & v_i \in \mathcal{B} \\ \max(z_{\text{sm}} + \max(z_q, z_f) + \frac{\max(d^i_{\mathcal{T}}, d^i_q, d^i_f)}{\tan\left(\frac{\alpha}{2}\right)}, z_{\min}) & v_i \in \mathcal{P}_\star \ \} \\ \max(z_{\text{sm}} + z_q + \frac{\max(d^i_{\mathcal{T}}, d^i_q)}{\tan\left(\frac{\alpha}{2}\right)}, z_{\min}) & \text{otherwise} \end{cases} \tag{3.1}$$

In Eq. 3.1, the choice of which vertices to use to form $\mathcal{P}_\star$ is arbitrary, however one practical heuristic is to preference the top face vertices for this role by selecting the vantage point vertex with the largest z-value for each polyhedron. Lastly, it should be noted that all distances considered are between projections of the vertices into the xy plane.

3.1.5 VWSGA Coverage Guarantees

The VWSGA provides an upper bound on the number of elements in the covering vantage point set, denoted N. Let $n_{\mathcal{B}}$ be the number of boundary vertices, k be the number of polyhedra present in the map, and $n_1 ... n_k$ be the number of vertices of

each bottom polygon. Then N is bounded by:

$$N \leq \frac{n_{\mathcal{B}} + 2n_1 + \cdot + 2n_k + 2k}{3} \tag{3.2}$$

The VWSGA's coverage guarantee is as follows. By construction, every triangle from the triangulation has a vantage point at one of its vertices, and every point along the boundary belongs to a triangle. The drone altitudes selected from the first case of Eq. 3.1 ($v_i \in \mathcal{B}$) ensure any point in the flat areas of the map is visible from at least one vantage point. Vantage point altitudes selected via the second case in Eq. 3.1 ($v_i \in \mathcal{B}$) guarantee that, for each polyhedron, one vantage point will be able to view the entire top face. Additionally, for each side quadrilateral face, there is at least one vantage point assigned to one of its vertices by construction, and the third case in Eq. 3.1 guarantees any point on the side face is visible.

3.2 VWSGA+: A Practical Improvement to the VWSGA

While originally intended to provide complete aerial coverage over complex uneven terrain for surveillance, the VWSGA and associated terrain model are well-suited for other applications, such as structural inspection and disaster response damage assessment, in which case the polyhedron could be used to model buildings and urban structures. Another interesting application, and the primary focus of this dissertation, is UAV-assisted wilderness search and rescue, in which case the polyhedron are used to model terrain and vegetation occlusion. However, there are certain limitations which must be addressed before the VWSGA can be effectively used for this purpose.

In order to guarantee the vantage points set is indeed a covering set for the geometric proxy, the VWSGA makes a key assumption which limits its practicality. Namely, the altitudes prescribed by the VWSGA are merely assumed to fall below some maximum altitude limit, $z_{\max}$. This lack of an upper altitude bound enforcement can lead to impractically high altitudes depending on the geometry of the underlying environment, which, in real-world UAV missions, would violate regulatory, opera-

tional, and/or technical altitude restrictions. Thus, adding an upper bound to the VWSGA's prescribed altitudes would make it practical for real-world use in generating occlusion-aware waypoint missions. However, simply capping the altitude of a vantage point to a desired $z_{\max}$ reduces the area which it can observe and, therefore, would invalidate the VWSGA's coverage guarantee. To compensate for the loss in visibility and recover the coverage guarantee, additional vantage points are required, however it is not immediately obvious where these new vantage points should be placed, nor how many may be required.

Depending on the geometry of the underlying terrain and triangulation results, Eq. 3.1 can yield a large range of altitude values. To bound the output of Eq. 3.1, we propose a method, referred to as VWSGA+, based on Steiner points, which are additional vertices used many computational geometry problems to improve the quality of the solution [63]. In this context, we use Steiner points to limit the allowable distances in the triangulation, as well as distances between side quadrilaterals and top faces. It should be noted that Steiner point additions are already used in conforming and constrained conforming Delaunay triangulation algorithms in order to limit the minimum angle and maximum area of each triangle, however these do not provide guarantees on the maximum triangle side length for an arbitrary set of input vertices.

Given a 2D graph representation of a geometric proxy, the placement of Steiner points is conducted in two steps. First, a triangulation side length constraint (denoted $\hat{d}$) is computed, and the graph is modified by adding intermediate vertices between segments of the bottom polygons, top polygons, and boundary, such that no segment exceeds $\hat{d}$. Second, a grid of Steiner points with grid spacing $\hat{d}$ in x and y is superimposed over the area-of-interest, and grid points which lie within the bottom face polygons are removed. The triangulation side length constraint is derived by considering the worst case output from Eq. 3.1, which can be seen in Fig. 3.1. Let $z_{\mathcal{P}} = \max(z_q, z_f)$ be the largest possible z-coordinate of any $v_i \in \mathcal{P}$, and $z_{\max}$ be the

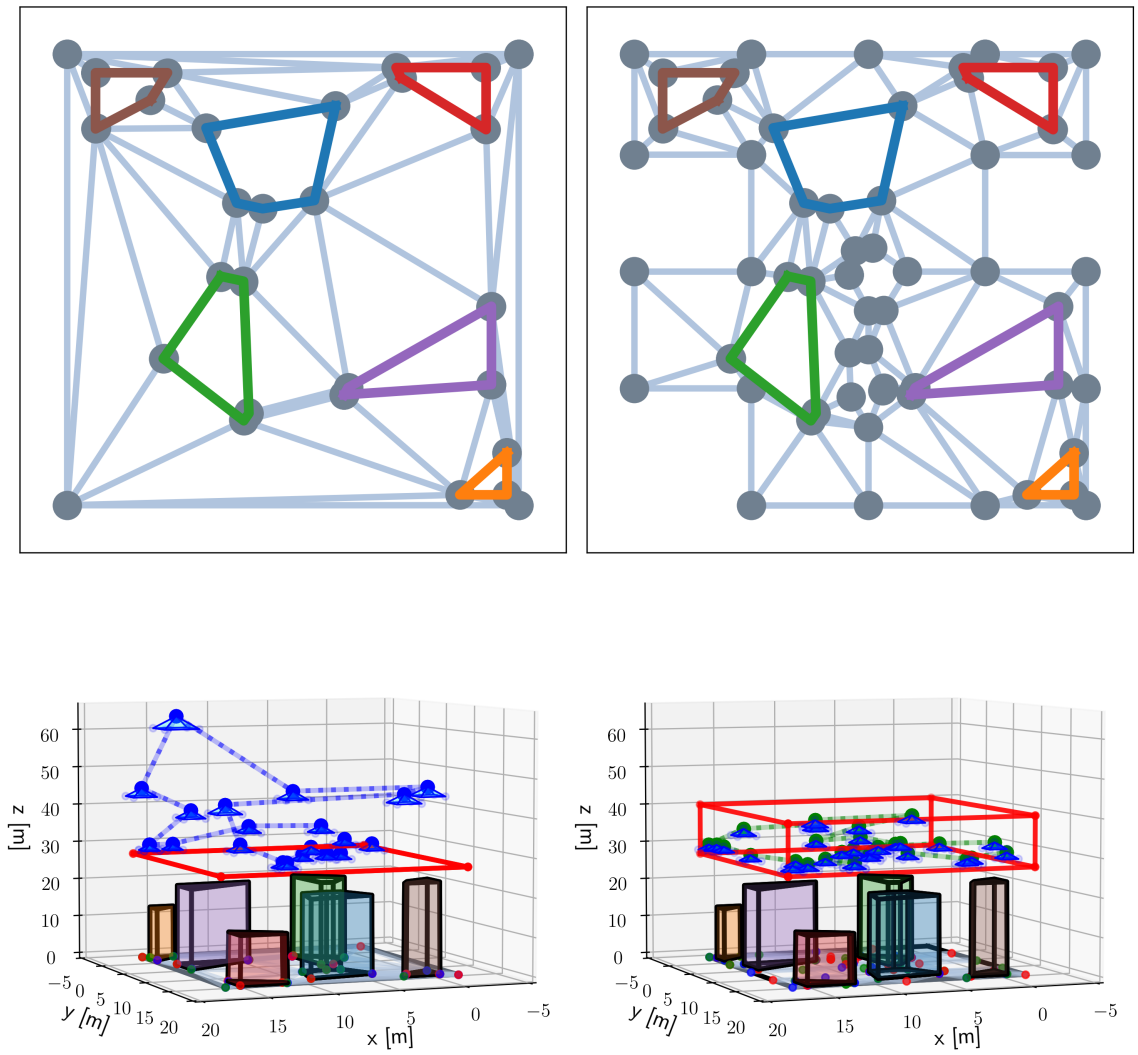


Figure 3.5: Triangulated graphs (top) and resulting trajectories (bottom) for the VWSGA (left) and VWSGA+ (right) for $z_{\min} = 30$m, $z_{\max} = 40$m, and $\alpha = 45°$. The addition of Steiner points in the VWSGA+ graph restricts the maximum triangle side length, bounding the altitude dimension of the vantage point set, as indicated by the red bounding box.

desired upper altitude bound on the vantage point set, then:

$$\hat{d} = (z_{\max} - z_{\text{sm}} - z_{\mathcal{P}}) \tan\left(\frac{\alpha}{2}\right) \tag{3.3}$$

The VWSGA+ Algorithm is outlined in Algo. 3. To illustrate the benefits of this algorithm, Fig. 3.5 provides a comparison of the graphs and trajectories generated using the original VWSGA and the VWSGA+ for a set of 6 polyhedra with $z_{\min}$=30m, $z_{\max}$=40m, and α=45°. The VWSGA trajectory features waypoint altitudes as high as 64m above ground level (AGL), which are driven primarily by the long triangle sides lengths between the corner points of the area-of-interest. These triangle lengths are restricted in the VWSGA+ graph, and as a result, the maximum waypoint altitude for the VWSGA+ trajectory is 38m, thus satisfying the $z_{\max}$ constraint.

3.2.1 Computational Complexity Analysis of VWSGA+

The algorithmic complexity of the VWSGA+ can be derived by examining the worst case complexity between the steps outlined in the *2D Coordinate Determination* procedure in Algo. 3. The complexity of the *cut and merge polys* step involves a comparison of vertex sets to find minimum distances between pairs in these sets, which is a type of nearest neighbor search. While described as a double *for* loop in Algo. 1, this step is implemented using a KD-Tree, which offers significant computational savings and has an established computational complexity of $\mathcal{O}(n \log n)$, where n is the total number of vertices in $\mathcal{M}$ [64]. The complexity of the Steiner point augmentation step is a function of the length and width of the area-of-interest, as well as $\hat{d}$, which is itself a function of the field-of-view and altitude constraints. The maximum number of points added by this augmentation, denoted by n_{S}, is $\Delta_x \times \Delta_y$, where $\Delta_x = \sqrt{2}(x_{\max} - x_{\min})/\hat{d}$ and $\Delta_y = \sqrt{2}(y_{\max} - y_{\min})/\hat{d}$. The augmentation involves checking these Steiner points against the polygons in the initial graph via a point-in-polygon algorithm, which has a complexity of $\mathcal{O}(n)$, but must be checked for each added point, resulting in a runtime complexity of $\mathcal{O}(n \times n_{\text{S}})$ [65]. For realistic values of FOV

Algorithm 3 Altitude-Constrained Vantage Waypoint Set Generation (VWSGA+)

procedure 2D COORDINATE DETERMINATION($\mathcal{M}$)

$G_{\mathrm{b}} \leftarrow$ construct bottom poly and boundary graph($\mathcal{M}$)

$\mathcal{G}_{\mathrm{m}} \leftarrow$ cut and merge polys($\mathcal{G}_{\mathrm{b}}$)

$\mathcal{G}_{\mathrm{s}} \leftarrow$ add Steiner points($\mathcal{G}_{\mathrm{m}}$)

$\mathcal{T} \leftarrow$ conforming constrained Delaunay triangulation($\mathcal{G}_{\mathrm{s}}$)

$\mathcal{T}_{\mathrm{e}} \leftarrow$ extend triangulation with top polygon vertices($\mathcal{T}, \mathcal{M}$)

$\mathcal{G}_{\mathrm{t}} \leftarrow$ three color($\mathcal{T}$)

$v^*_{\mathrm{2D}} \leftarrow$ get minimum color set coordinates($\mathcal{G}_{\mathrm{t}}$)

Return: v^*_{2D}, $\mathcal{G}_{\mathrm{t}}$

end procedure

procedure COMPUTE VANTAGE POINT ALTITUDES(v^*_{2D}, $\mathcal{G}_{\mathrm{t}}$, α, z_{sm}, Z_{min}, z_{max})

$v^*_{\mathrm{3D}} \leftarrow ()$

for $v_i \in v^*_{\mathrm{2D}}$ **do**

$z_q, z_f \leftarrow$ get maximum z-values($\mathcal{M}, v_i$)

$d^i_{\mathcal{T}}, d^i_q, d^i_f \leftarrow$ get maximum distances($\mathcal{M}, \mathcal{G}_{\mathrm{t}}, v_i$)

if v_i is a vertex of the polyhedra **then**

if top face not observed **then**

$z_i \leftarrow \max(z_{\mathrm{sm}} + z_q + \frac{\max(d^i_{\mathcal{T}}, d^i_q)}{\tan\left(\frac{\alpha}{2}\right)}, z_{\mathrm{min}})$

else top face already observed

$z_i \leftarrow \max(z_{\mathrm{sm}} + \max(z_q, z_f) + \frac{\max(d^i_{\mathcal{T}}, d^i_q, d^i_f)}{\tan\left(\frac{\alpha}{2}\right)}, z_{\mathrm{min}})$

end if

else v_i is a corner vertex

$z_i \leftarrow \max(\epsilon + \frac{d^i_{\mathcal{T}}}{\tan\left(\frac{\alpha}{2}\right)}, z_{\mathrm{min}})$

end if

$v^*_{\mathrm{3D}} \leftarrow \mathrm{append}([x_i, y_i, z_i])$

end for

end procedure

($\geq 15^{\circ}$) and altitude constraints ($\Delta_z \geq 5$m), the added computation from Steiner point augmentation can be limited or reduced by subdividing the choice of area into smaller problems. The Constrained Delaunay Triangulation step has a complexity of $\mathcal{O}(n \log n \log m)$, where m is the number of non-intersecting diagonals connecting the n vertices, which is bounded by the constraint $m \leq 3n - 6$ [66]. Three coloring was implemented as described in Algo. 2, which has complexity $\mathcal{O}(n)$, as does the altitude selection step [62]. Thus, the dominate source of the VWSGA+'s computational complexity is the Steiner point augmentation, which can be a bottleneck depending on the constraint set that defines $\hat{d}$. However, as will be discussed in the next chapter, this algorithm is coupled with a waypoint sequencing step. Specifically, the waypoint sequenced is derived using the Lin-Kernighan method to solve the TSP, which has a computational complexity of $\mathrm{O}((n + n_{\mathrm{S}})^{2.2})$; therefore, the waypoint sequencing step is the true bottleneck for the entire VWSGA+ pipeline [42].

Chapter 4

ALTITUDE-CONSTRAINED COVERAGE PLANNING FOR OCCLUDED ENVIRONMENTS

This chapter describes the design, implementation, and simulated performance analysis of an altitude-constrained occlusion-aware coverage planner for UAV search and rescue. Section 4.1 provides a high-level system overview, including a description of the methods used for creating geometric proxies from LiDAR data, generating viewpoint sets from those proxies, and transforming the vantage point set into a sequenced waypoint mission. Section 4.2 then describes the experimental setup for a series of simulations in which the performance of this planner is compared against baselines, the analysis from these experiments are included.

4.1 Occlusion-Aware Coverage Planner Implementation

Figure 4.1 depicts the system diagram for the occlusion-aware coverage flight planner designed and built for this research. The goal of this effort was to create a pipeline which, given mission parameters, transforms a canopy height model (CHM) of an area-of-interest into an executable waypoint mission. The first step in this process involves creating a geometric proxy of the area-of-interest, which models the extents of individual trees or groups of dense foliage as simple 3D geometric shapes. The combination of this geometric proxy with desired altitude and field-of-view (FOV) constraints (α, $z_{\min}$, $z_{\max}$) is used as the input to the Vantage Point Solver.

The Vantage Point Solver (blue block in Fig. 4.1) processes these inputs into a graph and determines the set of viewpoints which provide complete coverage of the geometric proxy. The visit order of this vantage point set is computed by approximately

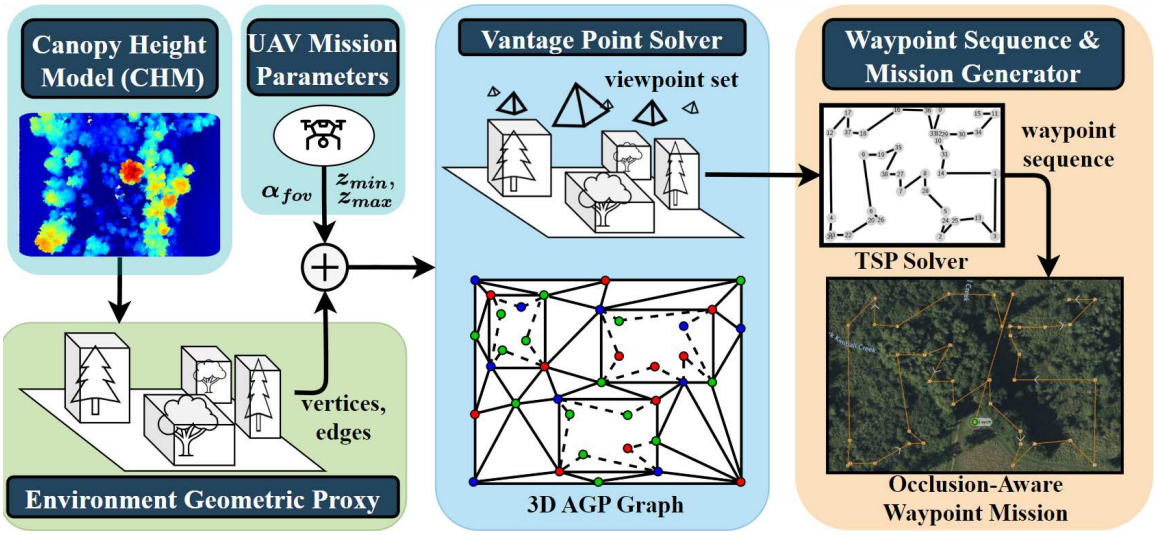


Figure 4.1: Occlusion-Aware Coverage Flight Planner System Diagram

solving a 3D metric Traveling Salesman Problem (TSP), and the full solution (including launch and land points, and trajectory yaw) is processed into a standard waypoint file which can be ingested into any Ground Control Station (GCS) application which supports the MAVLINK protocol (orange block in Fig. 4.1). Full descriptions of these modules are provided in the next two sections.

4.1.1 LiDAR-derived Geometric Proxies for Viewpoint Generation

In wilderness search and rescue missions, the primary sources of occlusion are terrain and vegetation, and the success of any occlusion-aware path planner depends on how well these complex features are modeled. This section describes the method used in this research for creating an adapted version of the terrain model discussed in Sec. 3.1 specifically for modeling occlusion in a wilderness setting given some *a priori* information about the environment's terrain and vegetation. In particular, LiDAR point cloud data offers a dense 3D representation of the environment useful for determining the location and extents of trees and topography. This research assumes a LiDAR survey of the area is available.

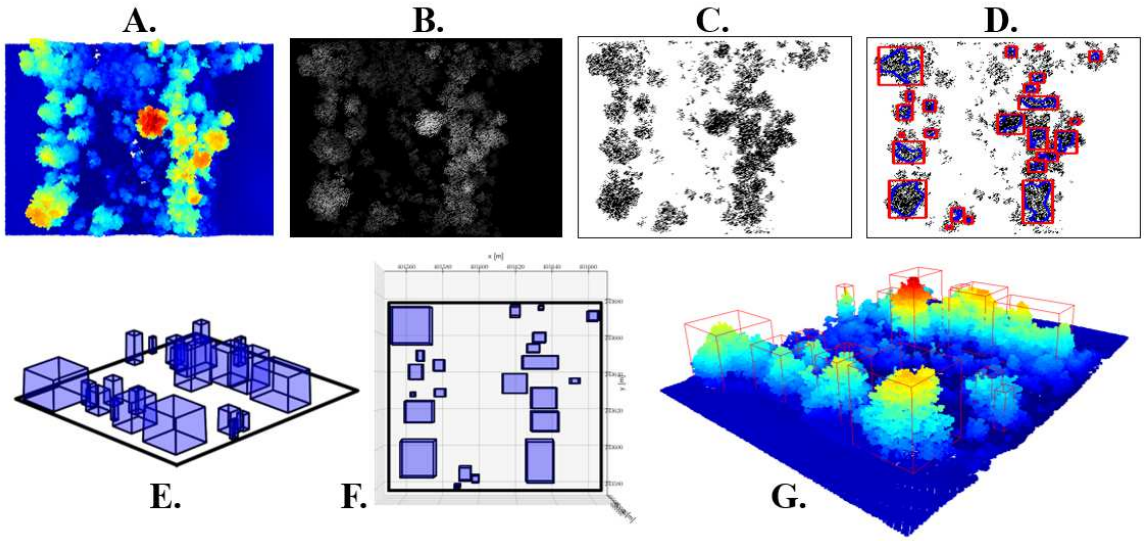


Figure 4.2: Polyhedron Map Generation from LiDAR. (a) point cloud visualization; (b) normalized canopy height model (CHM); (c) Grayscale CHM after morphological operations; (d) detected contours (blue) and bounding boxes (red); (e) 3D bounding boxes (Washington State Plane North coordinates); (f) overhead view of bounding boxes; (g) bounding boxes projected onto point cloud.

The availability of high-quality LiDAR data and derivative products continues to increase worldwide, as governments invest in surveys to support a variety of functions, including environmental and natural resource management, flood risk assessments, disaster response, land administration, and scientific research. Furthermore, LiDAR data maintains its accuracy over time because tree growth rates are generally slow enough to avoid significant changes in information related to occlusion, although there may be occasional exceptions (e.g., fallen trees or wildfire damage).

Figure 4.2 depicts the process used in this research for generating a geometric proxy of a real-world wilderness environment using LiDAR data from a 2017 LiDAR survey conducted in King County, Washington [67]. This process entails generating a normalized CHM (subfigure B in Fig. 4.2) from the point cloud data, processing it into a binary occlusion map (C in Fig. 4.2) using classical image filtering techniques,

performing contour detection (D in Fig. 4.2), and extracting the maximum tree heights to form 3D polyhedron (E-G in Fig. 4.2).

For simplicity, only the bounding boxes of detected contours are used for the polyhedron for real-world flight tests, however, it can be seen in subfigure D in Fig. 4.2 that the blue n-gonal polygons (inside the red bounding boxes) could be also be used for more precise modeling, and the number of vertices used to approximate the contour can be easily reduced using the Ramer-Douglas-Peucker algorithm [68].

The process which transform the normalized CHM into a binary map which can be used for contour detection is as follows. First a Gaussian blur is applied to the normalized CHM, followed by morphological closing (erosion followed by dilation) to remove noise and create separation between nearby trees, and applying an adaptive binary threshold using a Gaussian-weighted sum of the pixel neighborhood as the threshold value. Lastly, the maximum z-values within each contour bounding box are used to form the top face of the polyhedron, and the resulting 3D bounding box is projected from pixel coordinates to geodetic coordinates for use in the planner (G in Fig. 4.2). This order of operations and the parameters used for filtering were determined experimentally and found to produce serviceable geometric proxies, however, it should be noted that this process (like the majority of CHM-based ITDS methods) is not fully automated and may require a manual step of data-specific tuning for window size and threshold setting.

4.1.2 Vantage Point Solver

The Vantage Point Solver (blue block in Fig. 4.1) is a modified implementation of the VWSGA+ discussed in Sec. 3.2. The primary difference between this implementation and the Algo. 3 as described is that the former disregards the top face of the polyhedron. The reason for this is that, for the search and rescue application, covering only the side faces of the polyhedron is sufficient. Observing the top face of tree bounding boxes does not add value in terms of locating a human on the forest floor, and only

adds a penalty in the form of increased altitude for certain waypoints.

4.1.3 Waypoint Sequence Generation

Many open-source TSP solvers exist which include implementations of the algorithms described in 2.4. The research presented in this document leveraged the *python-tsp* library [69]. Given the size of the graphs ($\geq$ 500 in extreme cases) and number of simulations required for analysis, the *python-tsp* solver was configured with the 2-opt scheme for local search. The speed and efficiency this choice grants comes as the cost of accuracy and bias. In particular, there will be an inherent bias against larger vantage point sets due to the higher chance of getting trapped in local minima. This translates to producing less efficient (longer distance traveled) paths, on average, when compared to the vantage point sets with fewer vertices. Therefore, all else being equal, waypoint sequences generated via VWSGA+, which systematically add Steiner vertices to the problem, are less likely to be globally optimal compared to VWSGA sequences.

Ideally, this bias could be combated by using the worst-case performance guarantee provided by the Christofides-Serdyukov algorithm, which would bound all waypoint sequences to solutions that have, at most, 3/2 times the optimal cost [44]. However, run times for this algorithm are substantially worse compared to 2-opt, and deemed impractical for the graph sizes involved. The simulated annealing local search scheme was also considered, however this scheme would only partially address the bias in that it can still get trapped by local minima, and run times for this method were impractical for some graph sizes. As the results will show in Sec. 4.2, the use of 2-opt yielded reasonable waypoint sequences and the bias is not a significant factor which would change the conclusion. Once the waypoint sequence is determined, it is modified to include the launch and landing profiles, and the entire set of waypoints is converted into mission plans which can be uploaded into QGroundControl, or any GCS software application compatible with the MAVLINK protocol [70].

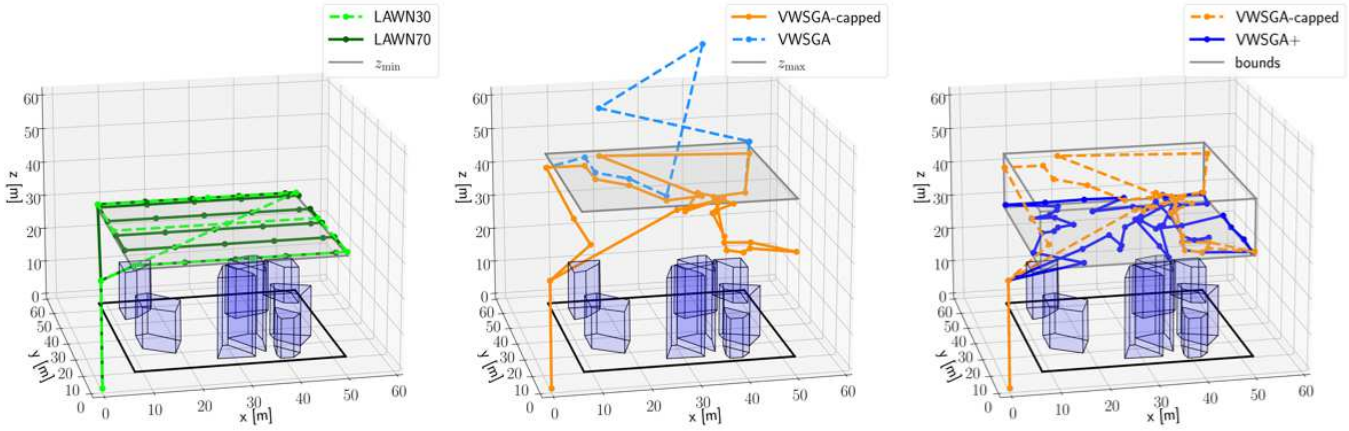


Figure 4.3: Example Simulated Trajectories. Lawnmower with 30% and 70% overlap (left), VWSGA and VWSGA-capped (right), and VWSGA-capped with VWSGA+.

4.2 Simulated Results and Analysis

The proposed altitude-constrained occlusion-aware planner was implemented in a simulated environment. Experiments were conducted to characterize the performance and understand the limitations of this planning method. Baseline waypoint missions were generated for comparison, including variants of the simple lawnmower pattern, and variants of waypoint missions generated with the original VWSGA algorithm. This section details the results and analysis from these experiments. These experiments indicate the VWSGA+ method provides guaranteed coverage of all polyhedron faces in the input map while respecting all field of view and position constraints, and the resulting flight performance characteristics are similar to a simple 70% overlapping lawnmower pattern.

.

Experimental Simulation Setup and Performance Measures

The performance of the VWSGA+ was evaluated in simulation across several configurations field of view angle (α) and altitude window ($\Delta_z = z_{\max} - z_{\min}$). Each Monte Carlo trial consisted of generating waypoint missions for a randomly generated map with a variable number of polyhedron (n_{poly}) between 4 and 10 polyhedrons, each with a variable number of sides between 3 and 6, and heights between 10m and 25m. The area-of-interest consisted of a fixed $40\times40\text{m}^2$ area, thus map density was variable between individual trials. Fifteen trials were conducted for each combination of altitude window parameter Δ_z, which ranged from 5m-30m in 5m increments, and fields of view angle, $\alpha = 40^\circ$ and $\alpha = 60^\circ$, resulting in n=180 total performance samples for each algorithm.

For each trial, waypoint missions were generated using the basic lawnmower pattern with a 30% FOV overlap (LAWN30), a lawnmower with a 70% FOV overlap (LAWN70), the original VWSGA (VWSGA), the VWSGA with altitudes artificially limited to $z_{\max}$ (VWSGA-capped), and the VWSGA+. The VWSGA-capped data allows for a fair comparison with respect to coverage in the context of an upper altitude constraint. Additionally, full trajectories for these waypoint missions were generated and simulated using quadrotor dynamics with a simple position controller in order to collect performance measures for simulated thrust required, distance traveled, and mission time. The trajectories generated all shared common takeoff and landing profiles, and examples of these trajectories are depicted in Fig. 4.3. As an additional measure of the efficiency, the empirically-derived comprehensive energy consumption model described in [71] was employed. This model takes into account the differences in energy consumption for different flying modes, such as hovering, horizontal flight, vertical upward/downward flight, as well as conventional factors which impact battery usage, such as payload weight, wind and commanded airspeed.

Coverage was measured via brute force computation, performing visibility testing

(ray casting and intersection testing) between each vantage point with every vertex in the map within that vantage point's visible cone as defined by α (see Fig. 3.1). A polyhedron face was counted as being seen by a vantage point all of its vertices were visible, and the union of faces seen by all vantage points in the set represents the total number of faces covered for that waypoint mission, $\bar{f}_{\text{seen}}$. The total number of visible faces, which excludes bottom and top faces, is denoted as $\bar{f}_{\text{possible}}$. Since the VWSGA and VWSGA+ also guarantee coverage of points between the polyhedra and the boundary of the area-of-interest, the visibility of these vertices are also included in the coverage metric, where the number of boundary vertices seen, and total number possible are denoted $\bar{v}_{\text{seen}}$ and $\bar{v}_{\text{possible}}$, respectively (for the VWSGA+, this includes the additional Steiner vertices which are not part of any polyhedron). The coverage percentage is then given by:

$$m_{\text{coverage}} = \frac{\bar{f}_{\text{seen}} + \bar{v}_{\text{seen}}}{\bar{f}_{\text{possible}} + \bar{v}_{\text{possible}}} \times 100 \tag{4.1}$$

4.2.1 Simulated Performance Analysis: Coverage

Figure 4.4 displays the coverage statistics resulting from the simulation described in Section 4.2. The top row shows coverage performance as a function of altitude window for $\alpha = 40°$ (top left) and $\alpha = 60°$ (top right). The bottom row shows the combined coverage statistics for both fields of view, and the bottom right plot depicts the total coverage performance for each α across all values of Δ_z. Of note, the VWSGA+ achieves perfect coverage in all cases, which supports the claim that our method retains a coverage guarantee within the box constraint defined by the area-of-interest boundary, $z_{\min}$ and $z_{\max}$. As expected, the VWSGA-capped achieves less-than-perfect coverage, and its performance is poorest when Δ_z is small, and increases as Δ_z increases and the altitude window widens. The average coverage of the lawnmower patterns never exceed that of the VWSGA-capped nor the VWSGA+, and are relatively flat with respect to Δ_z because the altitude of these routes remained

constant for each FOV. Thus, the variances associated with the lawnmower results are strictly due to difference in the polyhedron map configurations, not changes in altitude window, whereas the variance in the VWSGA-capped results includes the effects of both. The VWSGA+ results have zero variance, indicating this method is not affected by the underlying map configuration or the magnitude Δ_z. Lastly, increasing α predictably improves coverage for the lawnmower patterns and VWSGA-capped, the latter of which is able to achieve coverage performance greater than 95% for $\Delta_z \geq 20$.

4.2.2 Simulated Performance Analysis: Flight Performance

Table 4.1 shows the mean and standard deviation of recorded performance measures for all simulations (top section), for only the trials with $\alpha = 40^\circ$ (middle section), and for only trials with $\alpha = 60^\circ$ (bottom section). Results obtained via the original VWSGA [36] are included for completeness and context, however as previously discussed, these results are not valid because the waypoints produced violate the upper altitude constraints; thus, the VWSGA-capped results should be used for comparison in its stead. Additionally, while the LAWN30 results would indicate best-in-class distances, mission time, energy, and thrust measures, this algorithm yielded worst-in-class coverage, averaging below 23% coverage across all trials and failing to achieve better than 45% coverage in any simulation, save for one lucky outlier. It would be misleading to include it as a valid contender for comparison with the other algorithms, so LAWN30 results are also only included for completeness.

Looking at the overall results in the top section of Table 4.1, the VWSGA-capped achieves lower average distance traveled, mission time, and thrust compared to VWSGA+ and LAWN70; however, the cost of these benefits is coverage, as evidenced by the 87% average coverage achieved by the VWSGA-capped versus the VWSGA+'s 100% in this category. In terms of energy consumption, LAWN70 achieves the lowest average at 37kJ, which is significantly lower than VWSGA-capped (53kJ) and

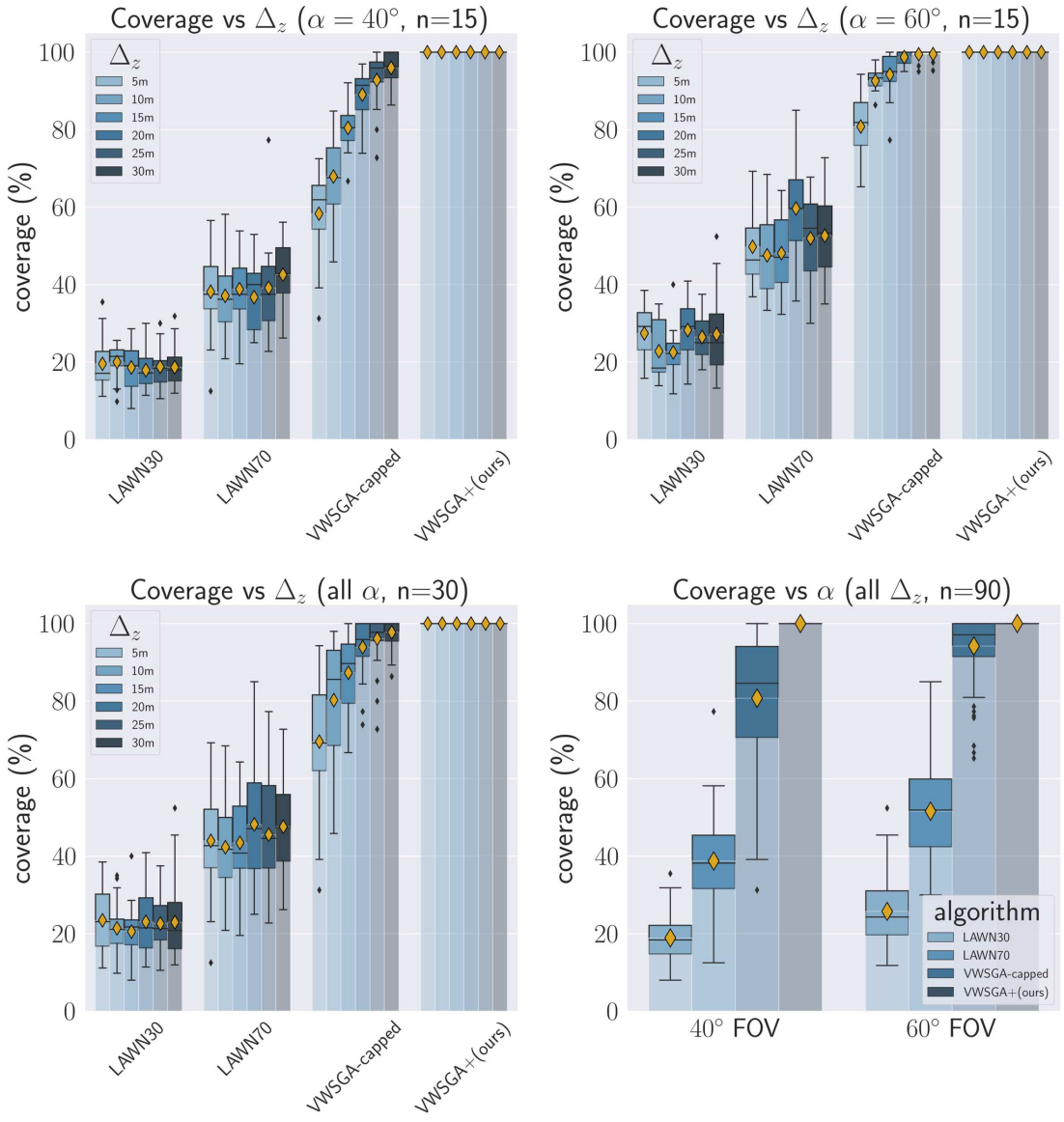


Figure 4.4: Coverage statistics as a function of altitude window (Δ_z) for 40° (top left) and 60° (top right) simulations. Combined statistics across all FOVs (bottom left) and all altitude windows (bottom right). Means indicated with gold diamond.

VWSGA+ (57kJ). This result can be attributed to the LAWN70's constant altitude, which allows it to avoid all altitude penalties in the energy consumption model.

4.2.3 High Variance in Thrust Measure

Although the energy consumption measure for LAWN70 is lowest across all trials, the simulated sum of thrust measure, which includes the acceleration and deceleration effects of stopping the quadrotor at each waypoint, is not the lowest. The VWSGA-capped achieves the lowest average thrust score because it requires the fewest waypoints, whereas the VWSGA+ requires the most thrust (on average) due to the added Steiner points, which creates more stop-and-go maneuvers in the simulation.

The variance in thrust score is significantly larger for the VWSGA+ in sections of Tab. 4.1. This variance is likely driven by the number of additional waypoints added by the VWSGA+ to meet the altitude constraints. The relationship between number of waypoints and thrust can be seen in Fig. 4.5. The bar and boxplots in this figure shows the relationship between the number of waypoints and altitude constraint. The number of waypoints required appears to grow exponentially as the altitude window decreases, and the combination of a tight altitude and FOV constraints can have a drastic effect on the number of waypoints required (e.g., $\alpha = 40°, \Delta_z$=5m). The correlation between thrust and the side length constraint (which is a function of altitude window) is shown in the lower half of Fig. 4.5. A cubic polynomial regression fit is used in this figure, which indicates a nonlinear relationship between side length constraint and thrust in the $\hat{d} \leq$ 10m regime, but shows a mostly constant, flat relationship beyond 10m.

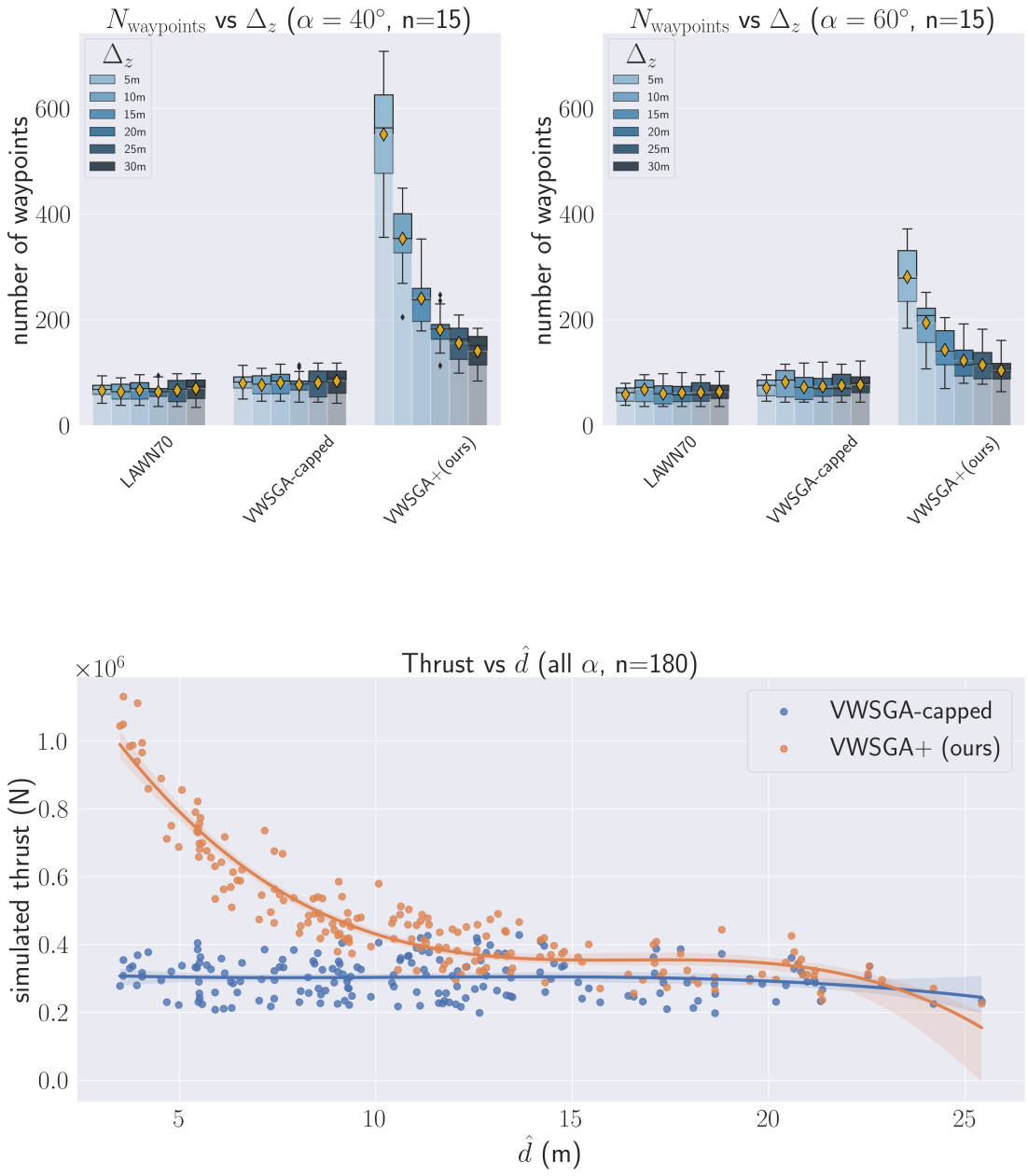


Figure 4.5: Relationship between altitude window and thrust. (Top) number of waypoints as a function of Δ_z for 40° and 60° simulations. (Bottom) Thrust as a function of side length constraint.

Table 4.1: Simulated Statistical Performance Comparison

algorithm	coverage (%, ↑)	distance (m, ↓)	time (s, ↓)	energy (kJ, ↓)	thrust (kN, ↓)
All trials (n=180)					
VWSGA [36]	100.00± 0.00	485.36 ± 89.91	140.86 ± 26.72	85.60 ± 17.74	338.64 ± 65.05
LAWN30	22.32 ± 7.64	332.39 ± 11.57	95.36 ± 7.49	28.049 ± 0.71	232.02 ± 19.70
LAWN70	45.16 ± 12.77	475.93 ± 44.12	179.96 ± 29.31	**36.879 ± 2.71**	447.07 ± 75.90
VWSGA-capped	87.47 ± 14.36	**401.92 ± 68.19**	**123.88 ± 23.06**	53.22± 12.81	**301.31 ± 57.15**
VWSGA+[ours]	**100.00 ± 0.00**	474.75 ± 85.96	197.30 ± 75.55	57.82 ± 10.59	484.428 ± 186.78
40° FOV (n=90)					
VWSGA [36]	100.00±0.00	540.63 ± 76.75	153.83 ± 24.31	97.32 ± 14.49	369.22 ± 59.76
LAWN30	18.89 ± 5.56	343.93 ± 0.00	102.83 ± 0.00	28.76 ±0.00	251.67 ± 0.00
LAWN70	38.73 ± 10.54	519.93 ± 0.00	209.27 ± 0.00	**39.58 ± 0.00**	522.73 ± 0.00
VWSGA-capped	80.73 ± 16.06	**425.41 ± 70.47**	**130.32 ± 23.47**	53.35 ± 14.27	**317.28 ± 58.16**
VWSGA+[ours]	**100.00 ± 0.00**	530.59 ± 76.01	233.19 ± 83.54	63.17 ± 9.66	572.79 ± 205.82
60° FOV (n=90)					
VWSGA [36]	100.00±0.00	378.43 ± 57.19	127.88 ± 22.46	73.88 ± 12.04	308.05 ± 55.22
LAWN30	25.75 ± 7.91	320.85 ± 0.00	87.8 ± 0.00	27.34± 0.00	212.38 ± 0.00
LAWN70	51.59 ± 11.55	431.93 ± 0.00	150.65 ± 0.00	**34.17 ± 0.00**	371.40 ± 0.00
VWSGA-capped	94.21 ± 8.05	430.10 ± 64.75	**117.44 ± 20.86**	53.07 ± 11.27	**285.33 ± 51.80**
VWSGA+[ours]	**100.00 ± 0.00**	**418.92 ± 52.70**	161.40 ± 43.52	52.47± 8.62	396.06± 109.61

Chapter 5

REAL-WORLD EXPERIMENTAL RESULTS AND ANALYSIS

A series of nine experiments featuring outdoor flight tests were conducted to validate the performance of the VWSGA+ and to characterize its real-world utility for mitigating occlusion in a search and rescue applications. This chapter describes these experiments in detail and provides a performance analysis and comparison to baseline methods, including the original VWSGA and lawnmower pattern.

5.1 Real-World Experiment Setup

Flight tests were conducted between July and August 2023 at Meadowbrook Farm Park located in North Bend, WA, which features a forested wilderness area. Objects and people were placed in the forested areas, as shown in Fig. 5.1. The selected locations featured varying degrees of occlusion from unoccluded (boxes C, D, F, and H in Fig. 5.1), partially occluded (boxes B and E), to heavily occluded (boxes A and G). The human shown in Box A was a teenager laying on a red-and-white striped blanket. The person in Box B was a partially-occluded adult in a sitting position. The person in Box C was an unoccluded child who was standing upright. The person in Box G was a partially-occluded adult in standing upright position. Objects included a pair of styrofoam UAV wings (box D), a white blanket (box F), and a back-and-white striped shower curtain (box H).

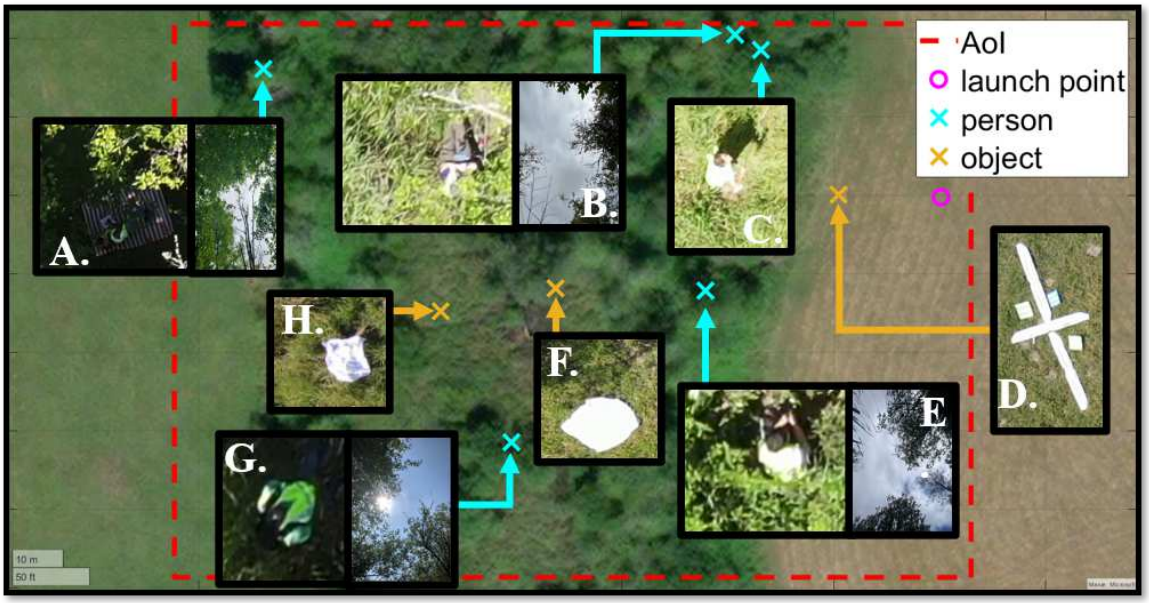


Figure 5.1: Object Placement for Experimental Scenarios. The positions of people (teal) and objects (orange) within the area-of-interest (red); these locations feature a range of easy (e.g., object D) to very difficult (person A) occlusion from aerial viewpoints. The view from these positions (looking straight up) is shown for boxes A, B, E and G.

Mission Profiles

Polyhedron maps of the trees in the area-of-interest were generated using the methods discussed in Ch. 3 using publicly available LiDAR point cloud data from a survey conducted in 2017 [67]. Vantage waypoint missions were then generated from these geometric proxies via the VWSGA and VWSGA+ algorithms for $\alpha = 85°$ and $\alpha = 46°$ in order to match the visual and thermal sensor lens configurations of the UAVs used for these experiments (hardware specifications are discussed further in Section5.1). As an additional baseline, a lawnmower pattern computed with the standard 70% overlap (the default setting in QGroundControl) for each FOV.

Figure 5.2 provides an overlay of the resulting waypoint missions for both FOVs, and the relevant parameters used to generate these missions are listed in Tab. 5.1.

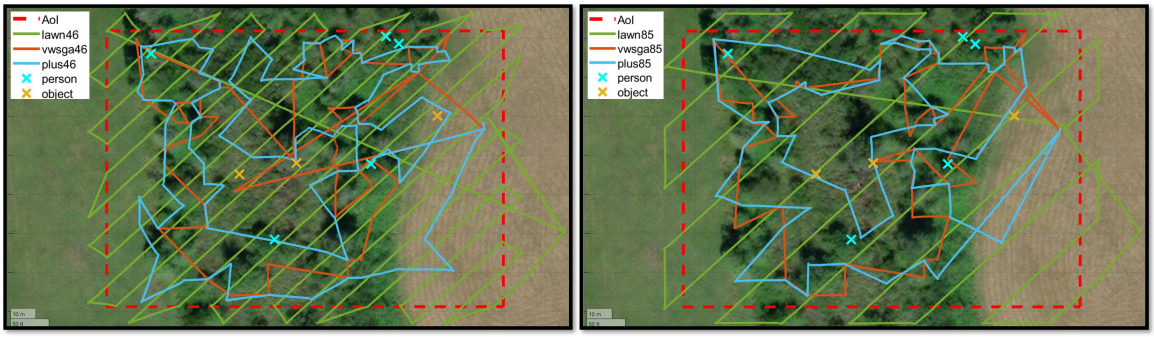


Figure 5.2: Waypoint missions for the 46° FOV (left) and 85° FOV (right) cases.

The main difference between the VWSGA+ profiles for the two fields of view was that a wider altitude window (higher $z_{\max}$) for the 46° case, in order to reduce the number of waypoints required for that mission. This was needed to ensure the waypoint mission was compatible with DJI's GCS software, which limits the number of waypoints allowed for a single mission. It is important to note that some of the waypoints produced by the original VWSGA for the 46° FOV were at altitudes higher than 245m AMSL (122m AGL), which is the FAA's legal altitude limit for small UAVs. These waypoints were therefore capped at 245m to comply with FAA regulations.

Table 5.1: Test Area Characteristics and Mission Parameters

Area-of-Interest		**Algorithm**		
characteristic	**value**	**parameter**	**46°**	**85°**
Area	17,066 m^2	$z_{\min}$	160m	160m
Max Tree Height	156.8 m	$z_{\max}$	195m	180m
Ground Elevation	126.9 m	ε	2.77m	2.77m
Latitude bounds	47.517240, 47.518224	z_{sm}	0.5m	0.5m
Longitude bounds	-121.805195, -121.807286	z_{max}(FAA)	245m	245m

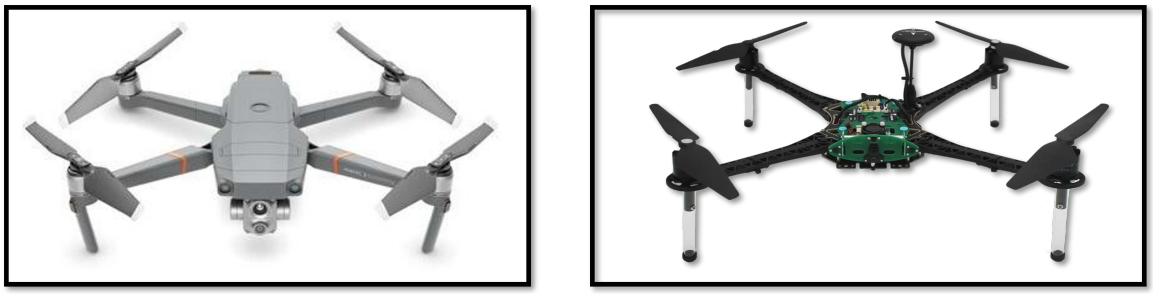


Figure 5.3: Flight Test Hardware. (Left) DJI Mavic 2 Enterprise Advanced; (right) ModalAI m500 Development Drone

Equipment and Hardware

Two quadrotor models were flown for these experiments, both of which are shown in Fig. 5.3. The majority of flights were conducted with a Mavic 2 Enterprise Advanced UAV equipped with a 48MP visual sensor with an 85° FOV and thermal sensor with 640×512 resolution and a 46° FOV. A second hardware configuration was flown, consisting of a ModalAI m500 Development Drone fitted with a 4k resolution downward facing visual sensor with an 85° FOV. Due to a hardware failure, the video quality for the ModalAI flights was significantly degraded and deemed unusable, so this video data is not included for analysis related to coverage. However, the ModalAI's flight performance-related data, such as mission time and battery usage, is valid and included in this analysis.

Coverage Performance Measures

To measure the coverage performance of the algorithms under test, a manual imagery analysis was conducted. This analysis consisted of reviewing the visual videos recorded during the 85° FOV missions, as well as the thermal videos recorded during the 46° FOV missions, and locating frames for each object and person placed in the forest, and drawing bounding boxes around these objects. The size of the resulting bounding boxes were then thresholded using Mesnik's criterion for detection,

recognition, and identification (DRI) to determine if an object or person was detectable [72]. The computed criteria under this standard given the relevant distances, camera specifications, lighting and atmospheric factors required 6 pixels across the target for detection (i.e., distinguishing an object from the background), 24 pixels across for recognition (distinguishing the object's class, i.e. human), and 50 across for identification (the ability to see details and features i.e., a female human wearing a hat).

Table 5.2 features the detectability matrix obtained from applying the Mesnik criteria to the results from manual imagery analysis and detection. Targets were classified by comparing the maximum bounding box dimension to the threshold values prescribed by the criteria. As this table shows, there were no cases of targets classified as undetectable (red), and in all cases, the visual modality achieved sufficiently high pixels-on-target to exceed the threshold for identifiability (green). Looking at the thermal video results, the VWSGA+ yielded the best result in that it was able to detect all targets, recognize three of the targets, and identify two. The detections from the LAWN70 thermal mission were the next best, with all targets detected, two of which were classified as recognizable, and two as identifiable. All targets were also detectable from the VWSGA videos, however due to this algorithm's bias towards high-altitude viewpoints, only one object was classified as recognizable and one as identifiable, and it should be noted that these latter two objects were physically the largest object in the target set. Zoomed in versions of the detected and cropped objects and people for each mission are provided as supplementary materials in Appendix C.

One caveat to the LAWN70 thermal mission's coverage result is that all targets were detectable, at least in part, due to the arbitrary choice of transect angle used to generate this pattern. A transect angle of 48° was chosen in order to reduce the amount of inefficient backtracking through the area-of-interest, and this choice was made without the knowledge of where the people and objects would be placed.

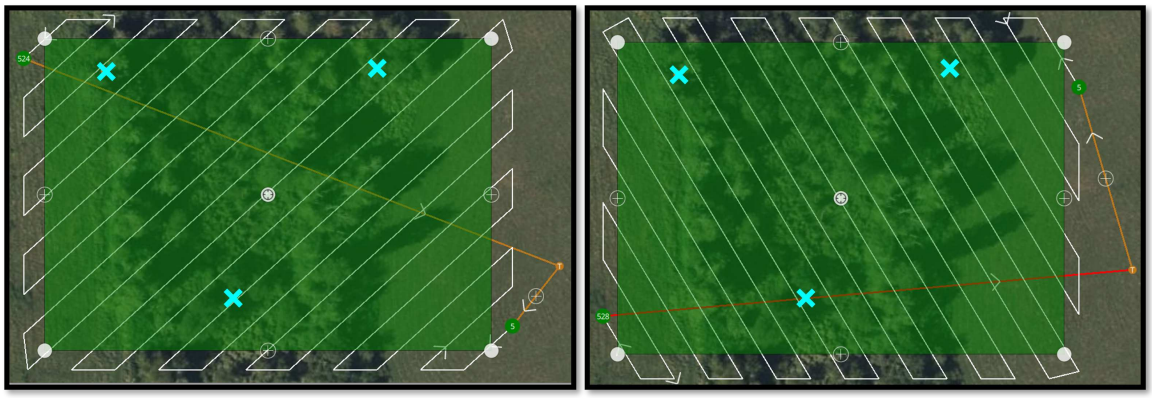


Figure 5.4: Lawnmower Sensitivity to Transect Angle. Lawnmowers generated with 48° (left) and 148° (right) transect angles, changing aspect views to ground points of interest (cyan markers).

Coincidentally, this decision enabled the UAV to fly directly over one of the most occluded positions (person A in Fig. 5.1), and this target in particular would have likely been missed had a different transect angle been selected. Figure 5.4 illustrates the lawnmower pattern's sensitivity to the transect angle parameter.

5.2 Flight Performance Analysis

Additional performance measures derived from experimental flight tests include distance traveled, total mission time, average speed, and battery usage percentage. While this analysis adds context to the overall performance of these algorithms, it is important to keep in mind that the VWSGA and VWSGA+ were not specifically designed to maximize efficiency, and thus coverage percentage is the most highly-weighted performance measure.

Flight Performance Summary

Table 5.3 summarizes the results across all performance measures, with best-in-class performance indicated in bold text. In both the 46° and 85° cases, the VWSGA+

Table 5.2: Detectability of Objects and People According to Mesnik's Criteria [72]

		Detection bounding box sizes from manual imagery analysis					
		46° FOV thermal detection box sizes (pixels)			85° FOV visual detection box sizes (pixels)		
Type	**difficulty**	**LAWN70**	**VWSGA**	**VWSGA+ (ours)**	**LAWN70**	**VWSGA**	**VWSGA+ (ours)**
Person A. (teenager)	hard	18x19	16x12	27x19	183x146	153x147	95x96
Person B. (adult)	med	16x18	17x23	23x24	73x95	87x71	45x67
Person C. (child)	hard	14x17	18x19	19x17	72x76	79x37	53x38
Object D. (wings)	easy	40x78	28x52	65x53	165x342	130x251	114x247
Person E. (adult)	med	18x24	13x18	18x23	73x63	44x68	60x33
Object F. (blanket)	easy	51x38	24x19	55x40	238x174	170x116	142x108
Person G. (adult)	hard	21x20	13x13	20x20	71x61	42x72	65x59
Object H. (curtain)	easy	33x36	16x16	32x31	142x125	87x93	81x83
Legend		**Mesnik's Criteria for Detection, Recognition, and Indentification (DRI)**					
			Undetectable	Detectable	Recognizeable	Identifiable	
min box dimensions (pixels)			<6	6x6	24x24	50x50	
Criterion computed with lens factor=0.82, atmospheric factor=1.0, low light noise factor=0.9, and Kell factor=0.7							

algorithm achieved the best overall flight performance with lower distance traveled, mission time, and battery usage, and these the VWSGA+'s performance advantages are significantly more pronounced in the 46° FOV case. Figures 5.5 and 5.6 show a comparison of altitude profiles for each mission. Intuitively, it takes more energy to gain altitude than it does to fly level or go down in altitude. The fact that VWSGA+ remains below 180m in altitude accounts for the better efficiency. Supplementary plots for comparing distance traveled, and battery usage flight test data are included in Appendix C.

Revisiting Tab. 5.3, while the lawnmower pattern achieved higher average flight speeds, VWSGA and VWSGA+ completed the 46° missions in 27.3% and 36.6% less time, respectively. In the 85°, the flight duration of the VWSGA+ was 9.2% shorter than the lawnmower, and 9.84% shorter than the VWSGA, which, interestingly, finished a 3 seconds later than the lawnmower.

Table 5.3: Real-World Experiment Performance Summary

	46° FOV (thermal)			85° FOV (visual)		
	lawn	vwsga	**vwsga+**	lawn	vwsga	**vwsga+**
distance (m)	3130	2141	**1720**	1792	1300	**1261**
time (mm:ss)	17:20	12:31	**10:59**	8:05	8:08	**7:20**
avg speed (m/s)	**2.92**	1.22	1.81	**3.69**	1.75	2.39
battery used (%)	66	48	**42**	31	31	**28**
detectable (%)	**100**	**100**	**100**	**100**	**100**	**100**
recognizable (%)	20	10	**30**	**100**	**100**	**100**
identifiable (%)	**20**	10	**20**	**100**	**100**	**100**

5.3 Limitations

While these results are encouraging and suggest VWSGA+ may provide sufficiently reliable and efficient coverage for wilderness search and rescue operations, there are limitations which should addressed. First, it should be understood that the VWSGA+ coverage guarantee only extends to the geometric proxy from which the viewpoints were derived. That is to say, this algorithm works well to the extent that a tree can be modeled with a polyhedron. Mismatches between the geometric proxy and the real world environment will inevitably exist, and these errors have the potential to cause certain regions in the area-of-interested to get missed. This is especially true when using dated LiDAR surveys as source data, in which case the real world occlusion picture can change in many ways, including fallen trees, new man-made construction, or drastic landscape changes from landslides.

The second limitation of this research is that ITDS is still not a completely solved problem, and the performance of these algorithms is degraded in broad-leaved forests, overlapping canopies, and closely spaced or clusters of trees with uniform heights.

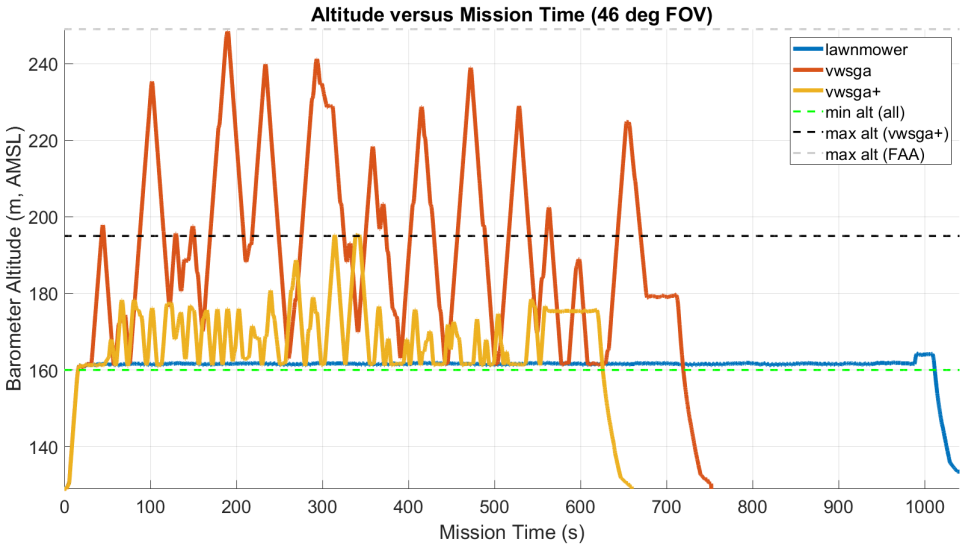


Figure 5.5: Flight Test Altitude Comparison for 46° FOV Missions

Error in tree detection is a likely source of error when using VWSGA+, as inaccurate tree locations will propagate into the geometric proxy and resulting viewpoint set. The method used for individual tree detection in this work were sufficient for the purpose of demonstrating the coverage algorithm's utility, but like the vast majority of image-based methods, it required some amount of manual tuning and parameter adjustment in order to create enough separation between clusters of trees for detection. More sophisticated and robust point-cloud methods have recently been proposed and could potentially lead towards a fully automated LiDAR-to-waypoint mission pipeline [47, 48, 50].

Lastly, the most significant limitation of the VWSGA+ is that the number of vertices required to generate an altitude-constrained coverage plan can grow exponentially as a function of the side length constraint, which is itself a function of the field-of-view and altitude parameters. In many cases this is not a problem, however, when the FOV and altitude window constraints are both tight, the side length constraint becomes small and the number of vertices required can become unmanageable.

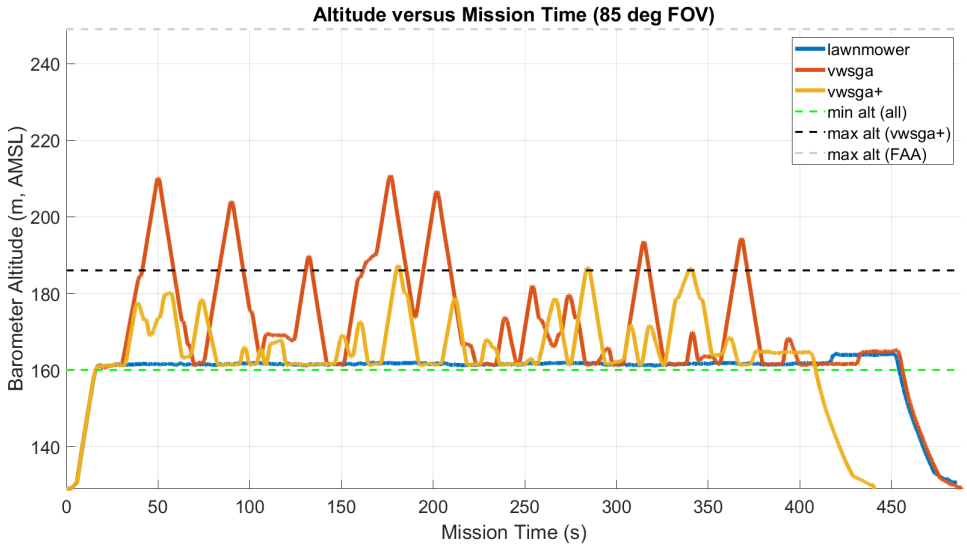


Figure 5.6: Flight Test Altitude Comparison for 85° FOV Missions

Simply put, this is the primary drawback to the VWSGA+ and is a limitation of the system. More investigation is needed to determine strategies for reducing the number of vertices required, perhaps using a divide-and-conquer approach or scheme for viewpoint consolidation.

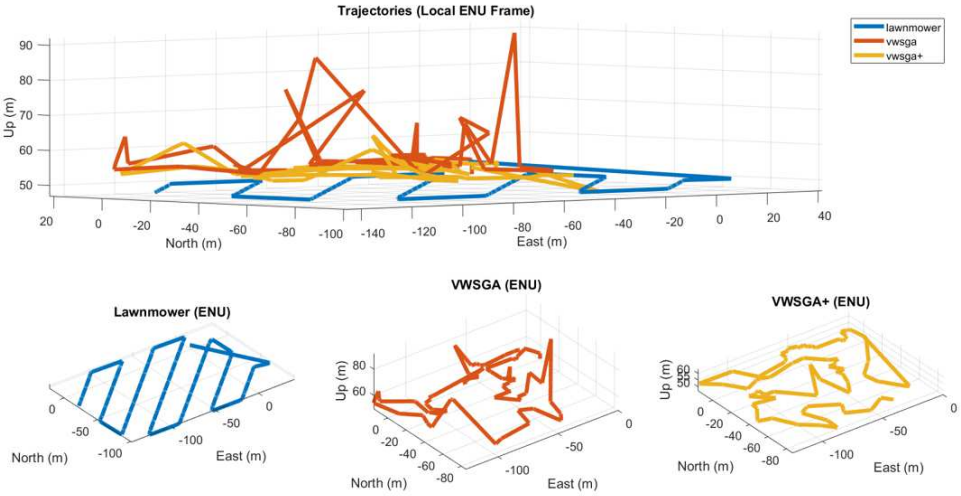


Figure 5.7: Local Flight Trajectories for 85° FOV Missions

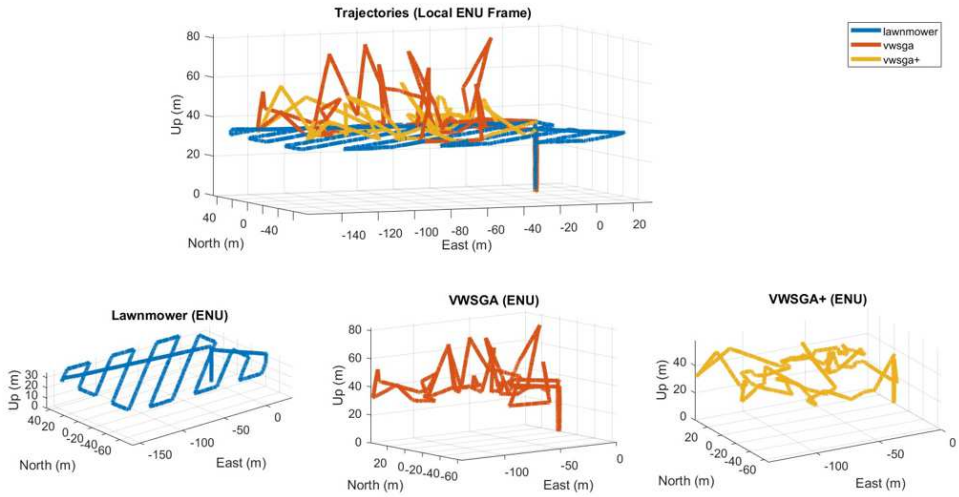


Figure 5.8: Local Flight Trajectories for 46° FOV Missions

Chapter 6
SUMMARY AND CONCLUSIONS

This dissertation investigated the autonomous coverage planning problem and addressed one of the primary challenges in this area: planning around environmental occlusion. The problem of occlusion-aware coverage is often framed as a combination of two subproblems, viewpoint generation and path planning, both of which belong to the NP-hard (or harder) complexity class. Existing mainstream coverage planners are insufficient for this task, resulting in the need for UAV operators to manually pilot these missions. The ability to generate reliable, occlusion-aware, autonomous waypoint missions would, at least in part, preclude this need, transforming a human-in-the-loop process into a human-on-the-loop process, and freeing the operator to focus on more important tasks, like imagery analysis or mission coordination.

To this end, this work proposed a new approach to occlusion-aware coverage planning, resulting in an altitude-constrained occlusion-aware coverage planner: the VWSGA+. The VWSGA+ coverage planner is a pipeline for generating autonomous waypoint missions from LiDAR-derived canopy height models (CHMs). The locations of individual trees, or clusters of trees, are extracted from the CHMs, and a geometric proxy of the area-of-interest is created, which models these trees as simple polyhedra. A covering viewpoint set that is bounded by a user-defined 3D geofence is derived from this proxy using the viewpoint generation method proposed in this research, which constitutes a practical and substantial improvement over an existing method. From this viewpoint set, a sequence of waypoints is constructed which minimizes travel distance, and the result is converted into an executable waypoint mission.

6.1 Conclusions from Experimental Results

The proposed altitude-constrained occlusion-aware planner was implemented and tested in simulation and real-world experiments. Simulated results indicate the proposed method achieves perfect coverage with respect to a geometric proxy from which it was derived, while respecting the altitude and area, effectively operating within the bounds of 3D goefence. Additionally, the simulated flight performance characteristics for waypoint missions generated with the proposed method were reasonable compared to baselines, but the proposed method was shown to produce an exponentially growing number of waypoints in cases with tight altitude and tight field-of-view constraints.

Real-world experiments featuring outdoor flight tests and mock search and rescue scenarios were conducted. In these experiments, the coverage performance of the proposed method was validated, and these flights yielded the best overall coverage, achieving viewpoints which captured images of all ten ground targets with sufficient detail for detection. Compared to baseline methods, the real-world flight performance characteristics suggest the VWSGA+ offers significant efficiency gains, as evidenced by the fact that in all cases, it used less battery, traveled less distance, and completed the mission faster.

6.2 Future Work

One of the more interesting ideas for future work in this area is the use of graph neural networks (GNNs) for viewpoint generation. GNNs are a suitable vehicle for Euclidean graph problems are used in a variety of other NP-hard combinatorial optimization problems, however the application to the 3D Art Gallery Problem has not been tried. Some preliminary formulations of a GNN architecture and training pipeline for performing the three-coloring task and altitude selection task were developed during the course of this research, however this work (included in Appendix A) is unfinished.

Another future work endeavor would be to replace the Traveling Salesman Problem (TSP) solver with a mutli-vehicle routing problem (VRP) solver to allow for multiple UAVs to simultaneously cover the area or, alternatively, a single UAV to cover a larger area with multiple feasible sorties, similar to the work of Shah [45]. A preliminary Satisfiability Modulo Theory (SMT) problem formulation adapted for the occlusion-aware coverage case was is in included in the Appendix B for the interested reader.

Given that the VWSGA+ relies on pre-existing LiDAR survey data, which may be years old, there are two research directions that could bear fruit. First, to address the mismatches between the geometric proxy and real-world, a supplementary online re-planning module could be developed. This online module would essentially modify the occlusion-aware graph or geometric proxy based on real-time sensor measurements, perhaps similar to how SLAM modules are used in model-free 3D scene reconstruction techniques. An additional aspect and future research direction is an investigation into other pre-existing sources which could be leveraged to derive suitable geometric proxies. For example, individual tree detection from satellite imagery coupled with shadow height analysis could potentially provide enough information for this purpose.

BIBLIOGRAPHY

[1] NPS. (2017) National Parks Service Annual SAR Dashboard. Accessed: 2022-03-12. [Online]. Available: https://arcg.is/fKbuS

[2] E. Sava, C. Twardy, R. Koester, and M. Sonwalkar, "Evaluating lost person behavior models," *Trans. GIS*, vol. 20, no. 1, pp. 38–53, 2016.

[3] T. Sharples. (2009) Get into trouble outdoors - who pays for the rescue? Accessed: 2022-03-12. [Online]. Available: http://content.time.com/time/nation/article/0,8599,1892621,00.html

[4] P. Rudol and P. Doherty, "Human body detection and geolocalization for UAV search and rescue missions," *2008 IEEE Aero. Conf., Vols 1-9*, pp. 3171–3178, 2008.

[5] D. Kovar. (2015) Uavs in sar - regulations, missions, requirements, and operations. Accessed: 2022-04-04. [Online]. Available: https://www.napsgfoundation.org/wp-content/uploads/2015/11/7a_UAV_SAR_Kovar_11142015.pdf

[6] M. A. Goodrich and A. C. Schultz, "Human-robot interaction: a survey," *Found. Trends Hum.-Comput. Interact*, vol. 1, no. 3, pp. 203–275, 2007.

[7] D. Broyles, C. R. Hayner, and K. Leung, "WiSARD: A labeled visual and thermal image dataset for wilderness search and rescue," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2022, pp. 9467–9474.

[8] T. G. Farr, P. A. Rosen, E. Caro, R. Crippen, R. Duren, S. Hensley, M. Kobrick, M. Paller, E. Rodriguez, L. Roth *et al.*, "The shuttle radar topography mission," *Reviews of geophysics*, vol. 45, no. 2, 2007.

[9] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, May 2006.

[10] T. M. Cabreira, C. D. Franco, P. R. Ferreira, and G. C. Buttazzo, "Energy-aware spiral coverage path planning for UAV photogrammetric applications," *IEEE Robot. Auton.*, vol. 3, no. 4, pp. 3662–3668, Oct. 2018.

[11] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Rob. Auton. Syst.*, vol. 61, pp. 1258–1276, 2013. [Online]. Available: https://dugi-doc.udg.edu/bitstream/handle/10256/9088/Survey-coverage-path-planning.pdf?sequence=1

[12] D. Li, X. Wang, and T. Sun, "Energy-optimal coverage path planning on topographic map for environment survey with unmanned aerial vehicles," *Electron. Lett.*, vol. 52, no. 9, pp. 699–701, 2016.

[13] L. Paull, C. Thibault, A. Nagaty, M. Seto, and H. Li, "Sensor-driven area coverage for an autonomous fixed-wing unmanned aerial vehicle," *IEEE Tran. Cyber.*, vol. 44, no. 9, pp. 1605–1618, 2014.

[14] C. S. Tan, R. Mohd-Mokhtar, and M. R. Arshad, "A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms," *IEEE Access*, vol. 9, pp. 119 310–119 342, 2021.

[15] R. Bähnemann, N. Lawrance, J. J. Chung, M. Pantic, R. Siegwart, and J. Nieto, "Revisiting boustrophedon coverage path planning as a generalized traveling salesman problem," *Field and Service Robotics*, Jul. 2019.

[16] Keld Helsgaun, "An effective implementation of the Lin-Kernighan traveling salesman heuristic," *Eur. J. Oper. Res.*, vol. 126, no. 1, pp. 106–130, 2000.

[17] S. O. Gharan, A. Saberi, and M. Singh, "A randomized rounding approach to the traveling salesman problem," *2011 IEEE 52nd FOCS*, pp. 550–559, 2011.

[18] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a large-scale traveling-salesman problem," *J. Oper. Res. Soc.*, vol. 2, no. 4, pp. 393–410, 1954.

[19] S. K. Ghosh, "Approximation algorithms for art gallery problems in polygons," *Discrete Appl. Math.*, vol. 158, no. 6, pp. 718–722, Mar. 2010.

[20] J. Marzal, "The three-dimensional art gallery problem and its solutions," Ph.D. dissertation, School of Information Technology Murdoch University, 2012.

[21] E. Semsch, M. Jakob, D. Pavlicek, and M. Pechoucek, "Autonomous uav surveillance in complex urban environments," in *2009 IEEE/WIC/ACM WA-IAT*, vol. 2. IEEE, 2009, pp. 82–85.

[22] M. Maboudi, M. Homaei, S. Song, S. Malihi, M. Saadatseresht, and M. Gerke, "A review on viewpoints and path-planning for uav-based 3d reconstruction," May 2022, accessed: 2023-03-12. [Online]. Available: https://arxiv.org/abs/2205.03716

[23] C. Connolly, "The determination of next best views," in *Proc. 1985 IEEE Int. Conf. Robot. Autom.*, vol. 2, Mar. 1985, pp. 432–435.

[24] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. 1997 IEEE ISCI*, Jul. 1997, pp. 146–151.

[25] N. Smith, N. Moehrle, M. Goesele, and W. Heidrich, "Aerial path planning for urban scene reconstruction: a continuous optimization method and benchmark," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 1–15, Dec. 2018.

[26] B. Hepp, M. Nießner, and O. Hilliges, "Plan3d: Viewpoint and trajectory optimization for aerial multi-view stereo reconstruction," *ACM Trans. Graph.*, vol. 38, no. 1, dec 2018. [Online]. Available: https://doi.org/10.1145/3233794

[27] M. Roberts, D. Dey, A. Truong, S. Sinha, S. Shah, A. Kapoor, P. Hanrahan, and N. Joshi, "Submodular trajectory optimization for aerial 3d scanning," in *Proceedings of ICCV*, 2017, pp. 5324–5333.

[28] A. Bircher, M. Kamel, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel, and R. Siegwart, "Three-dimensional coverage path planning via viewpoint resampling and tour optimization for aerial robots," *Auton. Robots*, vol. 40, no. 6, pp. 1059–1078, Aug. 2016.

[29] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, England: Cambridge University Press, Mar. 2004.

[30] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Rob. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011.

[31] M. Abrahamsen, A. Adamaszek, and T. Miltzow, "The art gallery problem is $\exists\mathbb{R}$-complete," *J. ACM*, vol. 69, no. 1, pp. 1–70, Dec. 2021.

[32] J.-M. Lien and N. M. Amato, "Approximate convex decomposition of polyhedra and its applications," *Comput. Aided Geom. Des.*, vol. 25, no. 7, pp. 503–522, Oct. 2008.

[33] M. N. Bygi and M. Ghodsi, "3d visibility graph," *Comp. Sci. App., Kuala Lampur*, 2007.

[34] Y. You, C. Cai, and Y. Wu, "3d visibility graph based motion planning and control," in *Proceedings of the 2019 5th Int. Conf. Robot. Artif. Intell.*, 2019, pp. 48–53.

[35] G. Juglan, "Solving the art gallery problem using gradient descent," Master's thesis, Utrecht University, 2022.

[36] A. V. Savkin and H. Huang, "Proactive deployment of aerial drones for coverage over very uneven terrains: A version of the 3D art gallery problem," *Sensors*, vol. 19, no. 6, Mar. 2019.

[37] J. Zhang and H. Huang, "Occlusion-Aware UAV path planning for reconnaissance and surveillance," *Drones*, vol. 5, no. 3, p. 98, Sep. 2021.

[38] Z. Li, Q. Chen, and V. Koltun, "Combinatorial optimization with graph convolutional networks and guided tree search," Oct. 2018.

[39] Y. Peng, B. Choi, and J. Xu, "Graph learning for combinatorial optimization: A survey of State-of-the-Art," *Data Science and Engineering*, vol. 6, no. 2, pp. 119–141, Jun. 2021.

[40] A. R. Karlin, N. Klein, and S. O. Gharan, "A (slightly) improved approximation algorithm for metric TSP," *53rd Annual ACM SIGACT*, Jul. 2020.

[41] F. Imeson and S. L. Smith, "An SMT-based approach to motion planning for multiple robots with complex constraints," *IEEE Trans. Rob.*, vol. 35, no. 3, pp. 669–684, Jun. 2019.

[42] D. S. Johnson and L. A. McGeoch, "The traveling salesman problem: A case study in local optimization," *Local search in combinatorial optimization*, vol. 1, no. 1, pp. 215–310, 1997.

[43] G. A. Croes, "A method for solving traveling-salesman problems," *Operations research*, vol. 6, no. 6, pp. 791–812, 1958.

[44] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," *Oper. Res. Forum*, vol. 3, no. 1, pp. 779–788, Mar. 2022.

[45] K. Shah, G. Ballard, A. Schmidt, and M. Schwager, "Multidrone aerial surveys of penguin colonies in antarctica," *Science Robotics*, Oct. 2020.

[46] K. Shah, "Safe large-scale aerial survey planning for multi-robot systems," Ph.D. dissertation, Stanford University, 2021.

[47] D. Tianyang, Z. Jian, G. Sibin, S. Ying, and F. Jing, "Single-tree detection in high-resolution remote-sensing images based on a cascade neural network," *ISPRS Int. J. Geo-Inf.*, vol. 7, no. 9, p. 367, Sep. 2018.

[48] M. Balsi, S. Esposito, P. Fallavollita, and C. Nardinocchi, "Single-tree detection in high-density lidar data from uav-based survey," *Eur. J. Remote Sens.*, vol. 51, no. 1, pp. 679–692, 2018.

[49] M. Münzinger, N. Prechtel, and M. Behnisch, "Mapping the urban forest in detail: From LiDAR point clouds to 3D tree models," *Urban For. Urban Greening*, vol. 74, p. 127637, Aug. 2022.

[50] Y. Pu, D. Xu, H. Wang, X. Li, and X. Xu, "A new strategy for individual tree detection and segmentation from leaf-on and leaf-off uav-lidar point clouds based on automatic detection of seed points," *Remote Sens.*, vol. 15, no. 6, p. 1619, 2023.

[51] C. Silva, C. Klauberg, M. Mohan, and B. Bright, "Lidar analysis in R and rlidar for forestry applications," *NR*, vol. 404, no. 504, pp. 1–90, 2018.

[52] J.-R. Roussel, D. Auty, N. C. Coops, P. Tompalski, T. R. Goodbody, A. Meador *et al.*, "lidr: An r package for analysis of airborne laser scanning (als) data," *Remote Sens. Environ.*, vol. 251, p. 112061, 2020.

[53] Qgroundcontrol - intuitive and powerful ground control station for the mavlink protocol. Accessed: 2023-03-12. [Online]. Available: http://qgroundcontrol.com/

[54] Pix4dmapper - the leading photogrammetry software for professional drone mapping. Accessed: 2023-03-12. [Online]. Available: https://www.pix4d.com/product/pix4dmapper-photogrammetry-software/

[55] Dji ground station pro. Accessed: 2023-03-12. [Online]. Available: https://www.dji.com/ground-station-pro

[56] Ardupilot apm mission planner 2. Accessed: 2023-03-12. [Online]. Available: https://ardupilot.org/planner2/

[57] Dronedeploy. Accessed: 2023-03-12. [Online]. Available: https://www.dronedeploy.com/

[58] Droneharmony. Accessed: 2023-03-12. [Online]. Available: https://www.droneharmony.com/

[59] Ugcs (universal ground control software). Accessed: 2023-03-12. [Online]. Available: https://www.ugcs.com/

[60] L. P. Chew, "Constrained delaunay triangulations," in *Proceedings of the third annual symposium on Computational geometry*, 1987, pp. 215–222.

[61] S. Goliaei and S. Jalili, "Optical graph 3-colorability," in *Optical Supercomputing*, S. Dolev and M. Oltean, Eds. Springer Berlin Heidelberg, 2011, pp. 16–22.

[62] A. A. Kooshesh and B. M. E. Moret, "Three-coloring the vertices of a triangulated simple polygon," *Pattern Recognit.*, vol. 25, no. 4, p. 443, Apr. 1992.

[63] M. Berg, O. Cheong, M. Kreveld, and M. Overmars, *Computational geometry: algorithms and applications*. Springer Science & Business Media, 2000.

[64] I. Wald and V. Havran, "On building fast kd-trees for ray tracing, and on doing that in o (n log n)," in *2006 IEEE Symp. on Interactive Ray Tracing*. IEEE, 2006, pp. 61–69.

[65] C.-W. Huang and T.-Y. Shih, "On the complexity of point-in-polygon algorithms," *Comput. Geosci.*, vol. 23, no. 1, pp. 109–118, Feb. 1997.

[66] T. C. Kao and D. M. Mount, "Incremental construction and dynamic maintenance of constrained delaunay triangulations," in *Proc. 4th Canad. Conf. Comput. Geom*, 1992, pp. 170–175.

[67] OCM Partners, "2016 - 2017 pslc lidar: King county, wa," 2022, accessed: 2021-05-07. [Online]. Available: https://www.fisheries.noaa.gov/inport/item/53388

[68] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica*, vol. 10, no. 2, pp. 112–122, 1973.

[69] F. S. Melo. (2023) python-tsp. Accessed: 2023-04-01. [Online]. Available: https://github.com/fillipe-gsm/python-tsp

[70] MAVLink Contributors. (2023) Mavlink file formats. Accessed: 2023-04-01. [Online]. Available: https://mavlink.io/en/file_formats/

[71] H. V. Abeywickrama, B. A. Jayawickrama, Y. He, and E. Dutkiewicz, "Comprehensive energy consumption model for unmanned aerial vehicles, based on empirical studies of battery performance," *IEEE Access*, vol. 6, pp. 58 383–58 394, 2018.

[72] Kintronics. (2022) Detection, recognition, identification: New criteria. Accessed: 2023-03-02. [Online]. Available: https://kintronics.com/detection-recognition-identification-new-criteria/

[73] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, Jul. 2017.

[74] R. Diestel, *Graph Theory*. 5th Ed., Springer, 2004.

[75] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. of the 22nd ACM SIGKDD*, 2016, pp. 855–864.

[76] H. Dai, B. Wang, R. Trivedi, and L. Song, "Learning graph representations with embedding propagation," in *Proc. of the 24th ACM SIGKDD*, 2018, pp. 2586–2595.

[77] J. Tang, M. Qu, M. Wang, and M. Zhang, "Line: Large-scale information network embedding," in *Proc. of the 24th Int. Conf. on WWW*, 2015, pp. 1067–1077.

[78] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. of the 22nd ACM SIGKDD*, 2016, pp. 1225–1234.

[79] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, Jan. 2020.

[80] Z. Zhang and M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," *Adv. Neural Inf. Process. Syst.*, May 2018.

[81] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. of the 31st Int. Confe. on NIPS*, 2017.

Appendix A

VIEWPOINT GENERATION WITH GRAPH NEURAL NETWORKS

This Appendix discusses a novel approach to the viewpoint generation problem using a graph neural network (GNN) to predict covering viewpoint sets. To our knowledge, this is the first application of GNNs to the 3D Art Gallery Problem (AGP), which was surprising given the numerous applications of GNNs to other NP-hard problems [39]. This appendix is organized, as follows. First, a background in the subfield of geometric deep learning and GNNs is provided, followed by a description of the proposed GNN architecture and initial experimental results.

A.1 Geometric Deep Learning and Graph Neural Networks

Geometric Deep Learning (GDL) is an umbrella term describing a unified theoretical and mathematical framework for studying neural network architectures and for extending deep neural models to domains that are not necessarily Euclidean (e.g., graphs and manifolds). This concept provides a blueprint for constructing deep learning architectures which exploit the *geometric priors* of the underlying data, resulting in highly-expressive function approximation while combating the effects of the so-called "curse of dimensionality." Furthermore, many of the most popular and successful deep learning architectures, including Convolutional Neural Networks (CNNs) and Transformers, can be derived from the GDL Blueprint; a fact which speaks to both its generality and practical applicability. This section covers relevant GDL concepts specifically for Graph Neural Networks (GNNs), however the interested reader is encouraged to review the works of Bronstein, et al., for more information [73].

A.1.1 Geometric Priors: Symmetry and Scale Separation

One of the core insights from GDL is that successful machine learning algorithms respect two fundamental principles: symmetry and scale separation, which are referred to as geometric priors. In this context, symmetry refers to a transformation that leaves an object unchanged or invariant, and scale separation refers to the ability to preserve characteristics of the data when transferring it to a coarser version of its domain. For many problems, these priors stem from the structure of the domain on which the data lives. For example, in image classification problems, object categories are unchanged by discrete shifts or rotations (symmetry), and images can be downsampled to a lower resolution while preserving important features (scale separation). Graph representations, on the other hand, require a different notion of symmetry, namely permutation invariance (permuting the input does not affect the output) or permutation equivariance (permuting the input results in a permuted output). More precisely, operations performed on graphs should not depend on the ordering of the nodes. Thus, the relevant symmetry group for graphs is the permutation group, whose elements consist of all possible orderings of the nodes in V.

To formalize the notion of an equivariant function, which is a key component of the GDL Blueprint, for graphs, consider a graph with n nodes, adjacency matrix A, and an $n \times d$ feature matrix X, the columns of which are d-dimensional features associated with an ordering of nodes specified by A. Note that applying a permutation matrix P to X would shift the columns of X, and this implies a new adjacency representation in the form of PAP^T. The node-wise function F is permutation equivariant if: $F(PX, PAP^T) = PF(X, A)$.

A.1.2 Graph Neural Networks (GNNs)

GNNs are a popular tool for understanding and analyzing relationships and patterns in highly complex data and/or systems modeled by graphs. This rise in popularity can

be partly attributed to the expressive power of graph representations, which can model intricate and subtle connections between molecules (micro-level), or relationships in large-scale social networks (macro-level). While, traditionally, the complexity of these representations has been problematic for machine learning algorithms, GNNs provide a scalable means for performing inference on graph data. Many popular deep learning architectures, including Convolutional Neural Networks (CNNs) and Transformers, can be understood as special cases of the GNN with some added structural component or choice of domain. The following section provides the requisite concepts from graph theory and geometric deep learning for understanding GNNs.

A.1.3 Graph Representations for Neural Networks

A graph is defined as a pair of sets, $G = (V, E)$, where the elements of the set V are nodes (also known as vertices) and the elements of E are edges, or connections between the nodes, represented by node pairs, such that $E \subseteq V \times V$ [74]. A graph may either be directed, meaning the edge (u, v) is not necessarily equivalent to (v, u), or undirected, in which case the order of the nodes does not matter when representing the edges of the graph. In practice, it is common to convert undirected graphs into directed graphs by adding the edges in both directions, which allows algorithms to be agnostic whether the input graph is directed or undirected.

To distinguish between nodes, each node is assigned a unique identifier, which we will refer to as the node index (typically, this is an integer). Edges can be represented by lists of node index pairs, or via an *adjacency matrix*. The adjacency matrix, A, of a graph with n nodes is an $n \times n$ matrix where the entries of A are given by:

$$A_{i,j} = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{A.1}$$

For consistency when using directed graphs, we will use the convention that the rows of A represent outgoing connections, and the columns of A represent incoming

connections. Note that the adjacency matrix of an undirected graph corresponds to the case when A is symmetric, or $A_{i,j} = A_{j,i}$.

Nodes and edges can also be endowed with additional attributes, or *features*, which capture characteristics of the problem being modeled. For instance, in a social network graph, participants in the network would be modeled as nodes with associated attributes like name, age, gender, and profile picture, and edges would model the relationships between these users, such as friendship or workplace affiliation. Another example is the graph representation for the TSP, in which case the node features are physical locations and the edges are weighted by the Euclidean distance between locations.

While node and edge attributes are powerful tools for modeling highly complex systems, they are often not directly suitable for applying machine learning techniques. In particular, attributes representing discrete or categorical variables, which cannot be compared with inequalities, are not directly usable for neural networks, which operate on multidimensional arrays. To alleviate this problem, attributes of this type are encoded into a vector form, known as an *embedding*.

Embeddings for Graph Representations

Embeddings, denoted as $x_i \ \forall \ v_i \in V$ for node features and $h_{i,j} \ \forall \ e_{i,j} \in E$ for edge features, serve two functions. First, they transform categorical attributes into continuous vector spaces, which have well-developed theory and mathematical properties and are amenable to a wider range of analysis techniques and algorithms (most importantly, neural networks). Second, embeddings effectively provide a compressed (lower-dimensional) feature representation, which is especially important for large graphs with millions or billions of nodes. Ideally, the embedding process will preserve some notion of similarity; that is, nodes which are similar in the graph representation should be close to each other in the embedding space (as measured by an inner product or distance function).

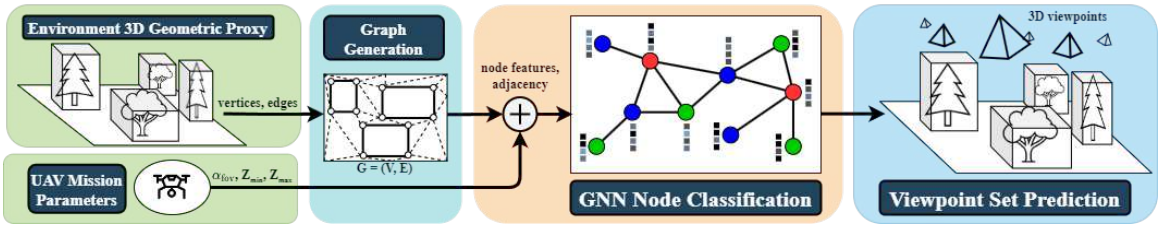


Figure A.1: GNN-based 3D Viewpoint Prediction Concept Diagram

Graph network embeddings can either be precomputed or be incorporated into an end-to-end learning scheme. When precomputed, the embeddings are learned independently with respect to any problem the GNN is attempting to solve, which makes the embeddings usable for a variety of problems or downstream tasks for the same graph. In end-to-end learning methods, the embeddings are generated as an intermediate step and influenced by downstream objectives, which can result in better task performance. Depending on the application, embeddings can be constructed at the node level, edge level, or graph/subgraph level. Two of the more popular methods are random walk-based (DeepWalk, node2vec, and struct2vec) and autoencoders (Large-scale Information Network Embedding (LINE) or Structural Deep Network Embedding (SDNE)) [75–78]. For node-level embedding where each node in the graph has m attributes, the resulting output of these methods is an $n \times m$ matrix X, called the node feature matrix, which is fed as an input into the GNN layers [39, 79].

A.2 Graph Neural Network (GNN)-based Viewpoint Generation

This section discussed a novel application of machine learning to the problem of viewpoint generation and discussed the design of a Graph Neural Network (GNN) architecture for this purpose. Specifically, given a terrain model, can we learn a viewpoint set which approximates a 3D AGP solution? Figure A.1 illustrates a conceptual viewpoint prediction scheme using a GNN. In the context of supervised learning, training examples for learning this task can be produced in the following way.

First, 3D polyhedra maps could be randomly or procedurally generated, which each map containing a random number of polyhedra, each with a randomized number of upper and lower vertices. The graph representation of this map, or its adjacency matrix, could then be used as input to a Graph Convolutional Network (GCN) [79], which has been successfully used to find heuristics for approximating the solution to combinatorial optimization problems, including the TSP. As of this writing, this approach has not been applied to the AGP.

The ground truth label could be generated by the Vantage Waypoint Generation algorithm, which operates on the same input graph. The learning task is then a node-level classification problem, where each node is classified as belonging to one of the three sets corresponding to the three-coloring algorithm. The output would be the set with the minimum number of elements, which corresponds to the minimum covering set of vantage waypoints. A commonly used loss function for node classification is cross-entropy loss, given by:

$$L_{CE} = -\sum_{i=1}^{n} t_i log(p_i) \tag{A.2}$$

In the above, n is the number of classes (three in this case: red, green, blue), t_i is the ground truth label for class i (output from Vantage Waypoint Generation), and p_i is the predicted class (softmaxed) probability [80].

A.2.1 GNN Pipeline Design

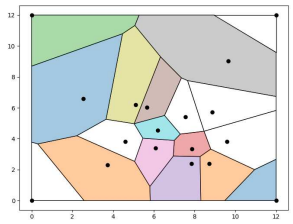
This section describes the GNN training pipeline used to learn covering viewpoint sets from a geometric terrain proxy. The pipeline is composed of two GNNs which perform distinct functions and are trained using different techniques. The purpose of the first network, referred to as G_1, is to predict the x, y coordinates of a covering viewpoint set given a triangulated planar line graph representation of a 3D polyhedra configuration. The second network, which will be referred to as G_2, predicts the appropriate viewing altitude of these viewpoints given the edge configuration and

relative distances between vertices in the graph. The following subsections describe the dataset generation and training methods, network architecture design, and implementation details of these two networks. Additionally, analytical and statistical results from Monte Carlo performance evaluation are provided.

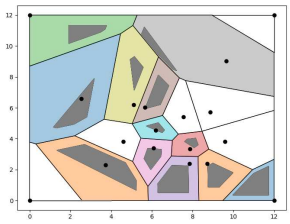
Dataset Generation

A dataset with over 40,000 graph instances was created to train the G_1 and G_2 models. Each graph instance in the dataset represents a randomly generated configuration of polyhedra with uniformly randomized characteristics, including: number of sides, width, height, and placement. The polyhedra are formed by connecting two regular polygons, a base polygon whose vertices are confined to the $z = 0$ plane, and a top face polygon, which has the same number of vertices as the base and its $z = 0$ projection is fully contained by the base (this geometric object is known as a *pyramidal frustum*). Constructing polyhedra in this way enables use of the Vantage Waypoint Set Generation algorithm, which assumes a 3D terrain model of this type and leverages the geometric properties of the polyhedra in order to compute a covering set.

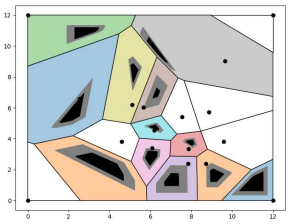
The generation and placement of polyhedra is accomplished in two steps. First, a Voronoi diagram is constructed from randomly drawn seed points, where the number of points used corresponds to the desired number of polyhedra, n_{poly}, plus a constant which specifies the number of unused Voronoi regions (this number affects the spacing and density of the 3D map). The Voronoi regions are then shuffled and the vertices from first n_{poly} regions are rescaled and processed to form polygons with the acceptable number of sides (by deleting/adding vertices, as needed) and are guaranteed to not overlap. These polygons form the base of the polyhedra, and a copy of these vertices are further down-scaled and coupled with a randomly drawn z-coordinate to form the top face of the polyhedra. The bottom and top faces are then joined with edges to form 3D polyhedra. Figure A.2 shows an example of the process with $n_{poly} = 12$. In the dataset, the number of polyhedron per map instance ranges from 3 to 15, each



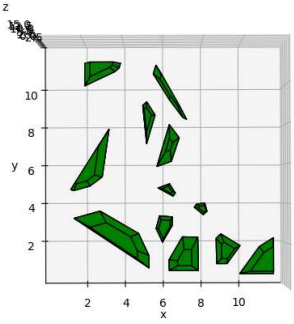
(a) Voronoi region selection



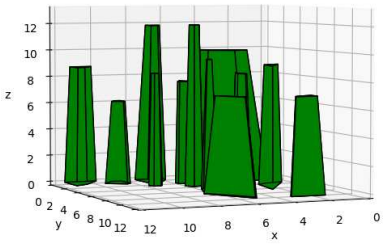
(b) Bottom polygons (grey)



(c) Top polygons (black)



(d) 3D map top view



(e) 3D map side view

Figure A.2: Voronoi-based 3D polyhedron generation process ($n_{poly} = 12$)

with a number of sides between 3 and 5, resulting in a variable number of vertices between 22 and 170. Note that, while the Vantage Waypoint Set Generation algorithm does not require the polyhedron to be convex, this generation method always results in convex shapes.

The graphs are post-processed with the Vantage Waypoint Set Generation algorithm, and therefore contain additional edges and vertices from the hole removal and constrained Delaunay triangulation procedures [60]. The computed waypoint solution is attached to the graph as an attribute. Node colors are converted into indices and used as labels for the node classification training task (i.e., red = 0, blue = 1, green = 2). For node features, each node is associated with spatial coordinates (normalized Euclidean x, y, and z coordinates) as well as a node degree feature (standardized by centering about the mean and scaling to unit variance). The spatial coordinates are also converted into distances associated with edges to form the edge features of the

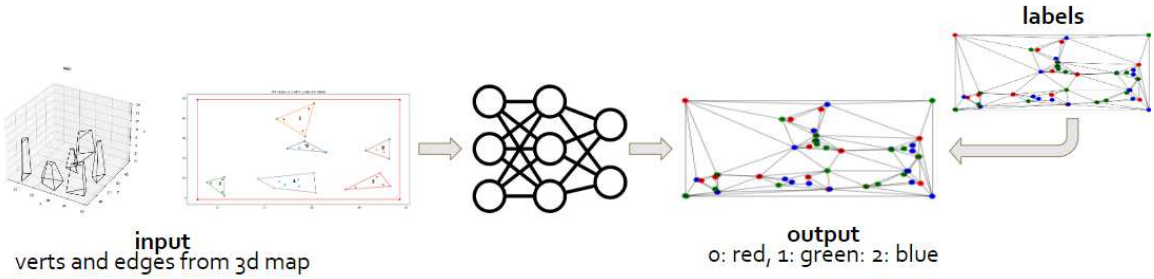


Figure A.3: G_1 node classification training diagram

dataset. Graph connectivity and adjacency information is provided in Coordinate List (commonly known as COO) format, which is a memory-efficient method for storing and constructing sparse matrices.

G_1: From 3D Polyhedra Map to 2D Waypoints

Figure A.3 diagrams the training scheme for G_1. The goal of G_1 is to quickly generate candidate 2D waypoint coordinates given a map of 3D polyhedra. To accomplish this task, the network classifies each node as belonging to one of three sets (red, green, or blue), and returns the set of nodes with the fewest number of elements. This is a node classification problem, where the predicted class determines how the z-coordinate for the waypoint is calculated. The design of the G_1 network was based on the GraphSAGE framework [81], and consists of SAGEConv message passing and linear layers with dropout regularization and ReLU activation functions. The SAGEConv message passing algorithm updates the embeddings of a node (denoted x_i for node i) by a weighted linear combination of its own embeddings with a weighted average of its neighbor's embeddings:

$$x_i' = W_1 x_i + W_2 \cdot \mathbf{mean}_{j \in N(i)} x_j \tag{A.3}$$

In the above equation, W_1 and W_2 are weights which are learned during the training process via back propagation of the loss. Thus, the network learns at the node-level

which nodes carry pertinent information for the prediction task, and which node's embeddings can be ignored or down-weighted. Figure A.3 shows the layout of G_1 forward pass, which shows how the node features are propagated though the network layers and hidden channels. The input to the network is an $n \times f$ tensor, where n is the number of nodes and f is the feature dimension. The SAGEConv layers are parameterized by h, which is the number of hidden channels, and with each layer the output dimension is reduced by a factor of two. The output layer is a linear function which outputs an $n \times c$ tensor, where n is the number of nodes and c is the number of classes (in this case, 3 for rgb). The softmax of this output tensor represents the class probability of each node, and the argmax is the prediction. In practice, a value of $h = 128$ hidden channels was experimentally determined to provide adequate performance in all cases.

A combination of cross entropy and cosine similarity loss was used for training the network to predict node color classes. Categorical cross entropy loss measures the dissimilarity between the predicted and true probability distributions (i.e., the models' predicted probabilities for each color and the true node color provided by the label). For categorical cross entropy, a mean reduction was applied in order to output a scalar loss. The function operates on the unnormalized logits (the output of the network prior to applying a softmax), and can be expressed as follows, where x_n is a 3×1 prediction of the probabilities [p(red), p(green), p(blue)], y_n is the true color of node n, and $x_{n,c}$ is the component of x_n corresponding to the class c:

$$l_{CE}(x_n, y_n) = -\log \frac{\exp(x_{n,y_n})}{\sum_{c=0}^{2} \exp(x_{n,c})} y_n \tag{A.4}$$

Cosine similarity is used to quantify the similarity of two vectors in inner product space. In the loss function, this metric is used to compute the average cosine similarity between a node's prediction and its neighbors' predictions. Typically, cosine similarity ranges from (-1,1), however when provided a softmaxed probability tensor as an input, the function outputs values ranging from 0 (for orthogonal vectors, dissimilar) to 1

(similar vectors pointing in the same direction). The model's softmax prediction can be thought of as a three-dimensional vector, where the axes are red, green, and blue. By virtue of the three-coloring algorithm, a node cannot have the same color as one of its neighbors, therefore adding this term to the loss function will penalize predictions that violate this rule. For a single node, the cosine similarity metric can be expressed as follows, where in this case x_n is the 3×1 softmaxed probability vector, and $x_{j \in N(n)}$ is the prediction for the neighbors of x_n:

$$l_{CS}(x_n) = \frac{x_n \cdot x_{j \in N(n)}}{\max(||x_n||_2 \cdot ||x_{j \in N(n)}||_2} \tag{A.5}$$

To output a scalar loss term, the cosine similarity of averaged across all nodes in the graph (or batch of graphs), denoted $\overline{l_{CS}}(x)$. In the final loss function, a tuning parameter, λ, was used to weight the cosine similarity penalty. For a training batch, the loss is given by the following, where x is the model output tensor for all nodes in the batch, and y is the corresponding labels.

$$l(x) = l_{CE}(x, y) + \lambda \cdot \overline{l_{CS}}(f_{softmax}(x)) \tag{A.6}$$

In practice, adding the cosine penalty with $\lambda = 0.5$ improved the accuracy and generalizability of G_1 network significantly. Figure A.4 shows the training loss for three training experiments, which are shown in different colors in the figure. Each experiment was conducted with $\lambda = 0$ (dotted traces) and $\lambda = 0.5$ (solid traces). When comparing the dotted and solid traces for each color, it is apparent the $\lambda = 0.5$ cases achieve lower steady state losses in less training steps when compared to their (dotted) counterparts. Furthermore, in Figure A.5, the average prediction accuracy for these experiments clearly shows higher achieved accuracy in less steps when the cosine similarity term is included.

G_2: From 2D to 3D Vantage Points

Figure A.6 diagrams the G_2 network. The goal of G_2 is to, given a set of 2D waypoints and a field of view, predict the appropriate z-coordinate which ensures all faces and

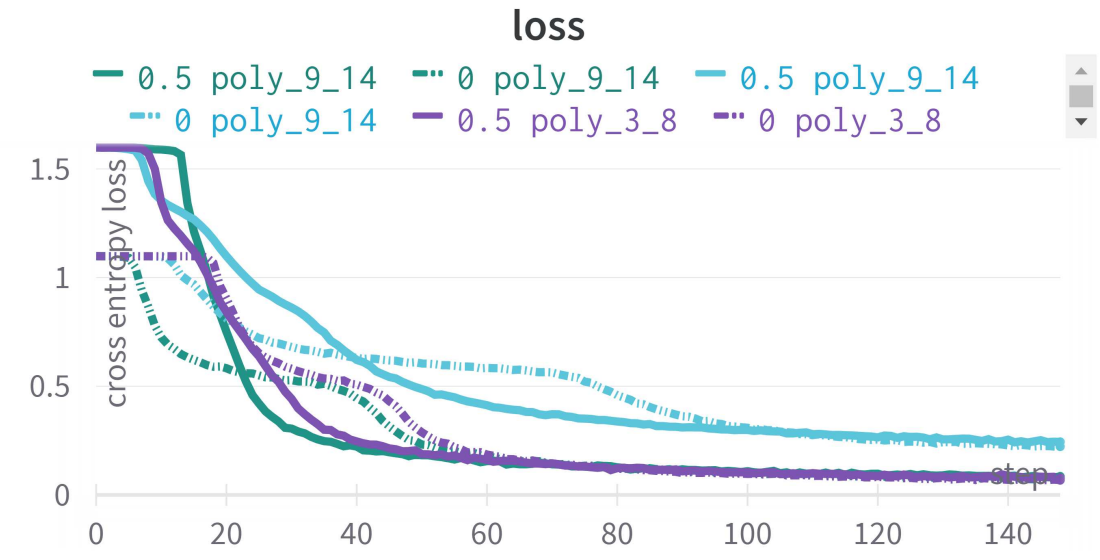


Figure A.4: G_1 Training Loss with and without Cosine Similarity Penalty

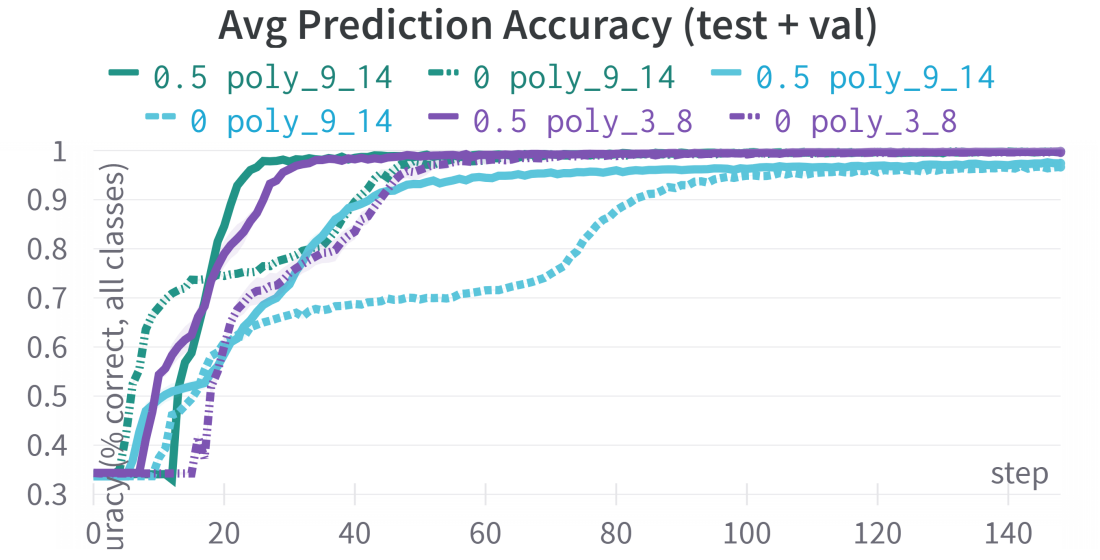


Figure A.5: G_1 Prediction accuracy with and without Cosine Similarity Penalty

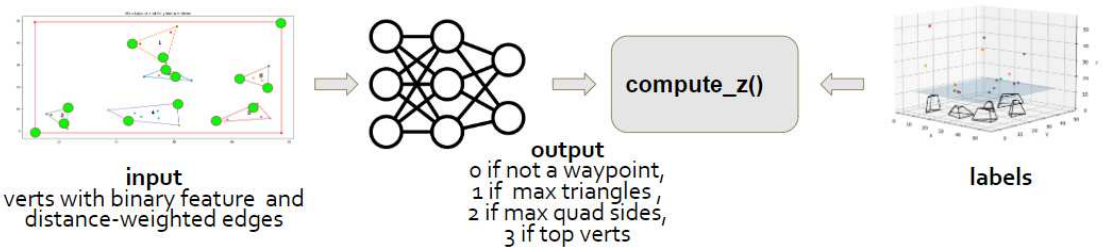


Figure A.6: G_2 node classification training diagram

vertices are seen (i.e., the z-value which approximately solves the 3D Art Gallery Problem). The input to the network is a graph representation of the 3D map, where the edges of the graph are weighted as a function of distance to connected nodes, and each vertex is assigned a binary feature signifying whether the node belongs to the minimum node set (i.e., the output of G_1). This is a node classification problem, where the predicted class determines how the z-coordinate for the waypoint is calculated. From the Vantage Waypoint Set Generation algorithm, once the 2D coordinates of the waypoint set is established, there are exactly three cases to consider when computing the z value (four, if we include the case where a node is not part of the waypoint set). The cases are as follows:

1. the node is not a waypoint
2. the node is a corner vertex
3. the node is a polyhedron vertex and the top face is observed
4. the node is a polyhedron vertex and the top face is not observed

These cases serve as labels which index into the set of functions used to compute the altitude value in the Vantage Waypoint Set Generation Algorithm (VWSGA). More specifically, referring back to the VWSGA's procedure for assigning the z-value for a waypoint outlined in Algo. 3, it can be seen that only a few factors, all of which are maximum distances, matter for the z-value assignment. For a Case 2 vertex,

the maximum triangle side length for which the vertex is an endpoint is used to compute the appropriate viewing altitude. For a Case 3 vertex, the relevant distance is a maximum of the triangle side lengths and the distance to all vertices of the two side quadrilaterals for which this point is a vertex. Lastly, for a Case 4 vertex, the maximum between triangle sides, side quadrilaterals, and distance to all top face vertices is used to compute the altitude. Since the input data to G_2 includes features which specify whether a node is a waypoint or not, and encodes the relevant distances as edge features, it is reasonable to expect that a graph neural network can learn the appropriate function to use for the z-value assignment. Future work will implement this second stage to create a full polyhedra map to viewpoint set pipeline.

Appendix B

OCCLUSION-AWARE SATISFIABILITY MODULO THEORIES (SMT)-BASED MULTI-AGENT COVERAGE PLANNING

In this Appendix, we extend the formulation provided in [45] to consider the occlusion-aware coverage problem, either for multiple UAVs or a single UAV via multiple sorties. Unlike the original formulation for the aerial survey application, there is no explicit requirement for camera frame overlap, but it is desired to encode poses that are aware of occlusion in the environment. The input graph is assumed to be generated in a way that accounts for occlusion (i.e., the output of the GNN or Vantage Waypoint Set Generation algorithm). Additionally, the original formulation computed the 2D coordinates and adjusted the z-value to keep the above ground level (AGL) height constant. In this formulation, the position of the UAV is encoded in 3D space[1].

Consider the following Metric TSP represented by the graph $G(V, E)$. The vertices of this graph ($v \in V$) represent poses which the camera must achieve in order to attain complete coverage. These nodes are mapped to 3D poses via the function $L : V \to \mathbb{R}^3$. The edges $e = (v_i, v_j)$ of the graph represent costs (distance traveled) associated with moving between v_i and v_j. Unlike the original formulation, which assumes G is a grid graph, we make no assumptions about the underlying structure of G.

Depending on the size of the area-of-interest and the allowable flight time, multiple sorties may be needed to achieve complete coverage. The problem is to assign each

[1]The reason for including an altitude state is to account for the fact that moving from low to high takes more energy than high to low; thus, the problem is not truly a Metric TSP. This dissertation only considers the metric TSP and leaves solving the asymmetric TSP for future work.

node v_i in the graph to one of K subsets $\{V^1, ..., V^K\}$, where K is the total number of flights required to cover the search area, and each subset forms the basis for a single coverage path, $V^k = \{v_1^k, ..., v_n^k, v_1^k\}$, where the first and last nodes are identical to make each path cyclic. In order to ensure each node is visited at least once, the sets are subject to the constraint:

$$\bigcup_{k=1}^{K} V^k = V \tag{B.1}$$

Next, it is desired to account for the time and distance traveled while transiting from the launch and recovery location to the search area and back. A launch and recovery location, or home point, is denoted $H^k \in \mathbf{H} \subset \mathbb{R}^3$, where $\mathbf{H}$ is the set of all possible home points. The home point may exist inside or outside the desired coverage area, but must be included in the solution path. The traveled distance for each coverage path over the search area (not including transit to/from the home point) is given by:

$$d_k = \sum_{i=1}^{n-1} \left\| L(v_{i+1}^k) - L(v_i^k) \right\| \tag{B.2}$$

To find which home point in $\mathbf{H}$ should be assigned to a given path, we compute the shortest distance between any home point and any point in the path V^k:

$$h_k = \underset{H^k \in H}{\arg\min} \left(\min_{v \in V^k} \left\| H^k - L(v^k) \right\| \right) \tag{B.3}$$

We assume the UAV travels between the home point and search area at a known speed v_h, and executes the coverage path at a speed v_s. Given these velocities, distances d_k and h_k, and the maximum flight time (T_{max}), we can compute the total flight time t_k and write the constraint specifying that each path must be completed within the time limit:

$$t_k = 2\frac{h_k}{v_h} + \frac{d_k}{v_s} < T_{max} \tag{B.4}$$

The problem is then to find a set of paths V^k and home point assignments H^k that minimize the total flight time:

$$\begin{aligned} \min \quad & \sum_{k=1}^{K} t_k \\ \text{s.t.} \quad & t_k < T_{max} \\ & \bigcup_{k=1}^{K} V^k = V \end{aligned} \tag{B.5}$$

In addition to minimizing the total flight time, it is desired to minimize the total number of flights K:

$$\begin{aligned} \min \quad & K \\ \text{s.t.} \quad & t_k < T_{max} \\ & \bigcup_{k=1}^{K} V^k = V \end{aligned} \tag{B.6}$$

Solving the optimization problems in Equations B.5 and B.6 for large graphs may lead to an exponential growth in the number of possible paths. To address this limitation, the graph G is decomposed into smaller graphs, denoted as $\hat{G}(\hat{V}, \hat{E})$. The minimum time (Eq B.5) is then solved for each subgraph, and adjacent grids are linked together using the cycle basis property of graphs, which preserves the cyclic nature of the combined subgraphs, ensuring the end result is a closed circuit [74].

For each tile, an SAT problem is formed using a one-hot encoding scheme, as follows. We first define the Boolean variable:

$$X_{\hat{v},\hat{t}} \in \mathcal{B} = \{0, 1\} \tag{B.7}$$

where $\hat{v} \in \hat{V} = \{\hat{v}_1 ... \hat{v}_N\}$ are one of N vertices in the subgraph and $X_{\hat{v},\hat{t}} = 1$ implies the UAV is at the vertex $\hat{v}$ at time step $\hat{t}$. With this representation, desired behaviors can be encoded by forming logical statements in either propositional (i.e., Boolean logic) or predicate (first-order logic, quantification) logic. The SAT problem is then to find an assignment (0 or 1) to each element of X such that the logical statements

evaluate to "true." In this formulation, the UAVs can only occupy a single vertex at any given time, which can be written as:

$$X_{\hat{v},\hat{t}} = 1 \ \forall \ \hat{t} \ \exists! \ \hat{v} \tag{B.8}$$

To encode the dynamics constraint that UAVs can only move to connected vertices, we allow the UAV to exist at a vertex $\hat{y}$ at time step $\hat{t}+1$ only if it was at vertex $\hat{v}$ at $\hat{t}$ and an edge exists between the two, denoted $\hat{v} \sim \hat{y} \implies (\hat{v}, \hat{y}) \in \hat{E}$:

$$\neg X_{\hat{v},\hat{t}} \vee_{\hat{v}\sim\hat{y}} X_{\hat{y},\hat{t}+1} \tag{B.9}$$

The constraint that every vertex is occupied at least once is given by:

$$\forall \ \hat{t} \mid X_{\hat{v},\hat{t}} = 1 \ \forall \ \hat{v} \tag{B.10}$$

After solving the SAT problem for each tile, the tiles are linked together in a way that preserves cycles and assigns a home point that minimizes backtracking through the search area. This can be accomplished by solving a mixed-integer linear program (MILP) graph partitioning problem, as constructed in [45].

Appendix C

SUPPLEMENTARY FLIGHT TEST DATA

This appendix provides additional data visualizations for the flight tests described in Chapter 5 of this dissertation. Figures C.1 and C.2 compare the total cumulative distance traveled versus mission time for the lawnmower, VWSGA, and VWSGA+ missions for the 46° and 85° FOV experiments, respectively. In both cases, the lawnmower travels significantly further than VWSGA and VWSGA+, and the VWSGA+ travels the least amount. It can also be observed from these figures that the VWSGA+ missions are completed significantly faster compared to VWSGA or lawnmower. Figures C.3 and C.4 compare the battery usage for the lawnmower, VWSGA, and VWSGA+ missions for the 46° and 85° FOV experiments, respectively. In both cases, the battery drain for the VWSGA+ was lowest. In the 46° case, the battery drain for VWSGA is significantly lower than lawnmower, however the VWSGA and lawnmower used the same amount of battery for the 85° case.

Figures C.5 through C.7 show cropped images of detected objects from the manual visual imagery analysis of video collected during the 85° missions. Figures C.8 to C.10 display the outcomes of the thermal video analysis for missions conducted at a 46° angle. As explained in Chapter 5, we utilized the sizes of the bounding boxes in these detections to assess whether an object met the criteria for detectability, recognizability, or identifiability based on an established standard. It's important to highlight that while some of the thermal detections may appear unclear and challenging to discern in these images, they were actually more visible in the video recordings due to the aircraft's movement.

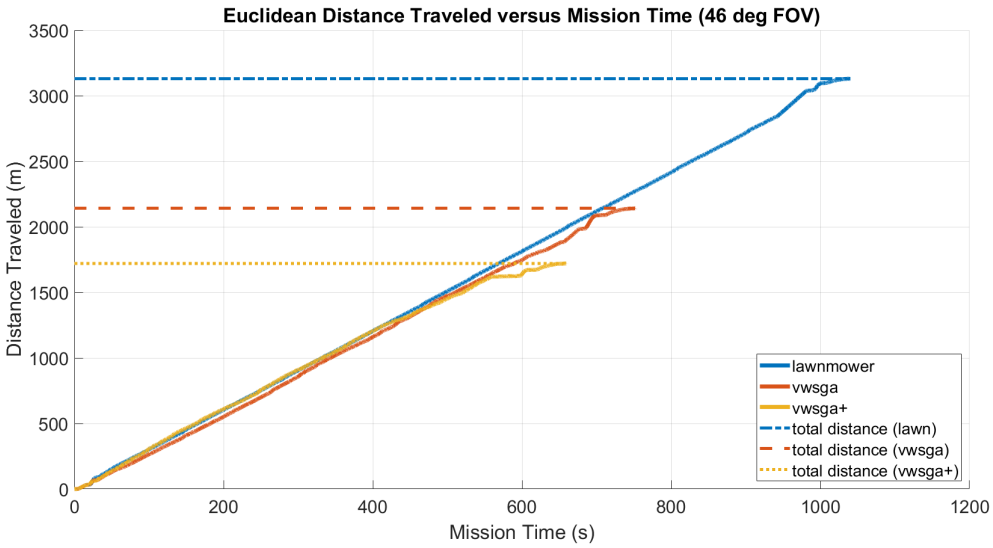


Figure C.1: Flight Test Distance Traveled Comparison for 46° FOV Missions

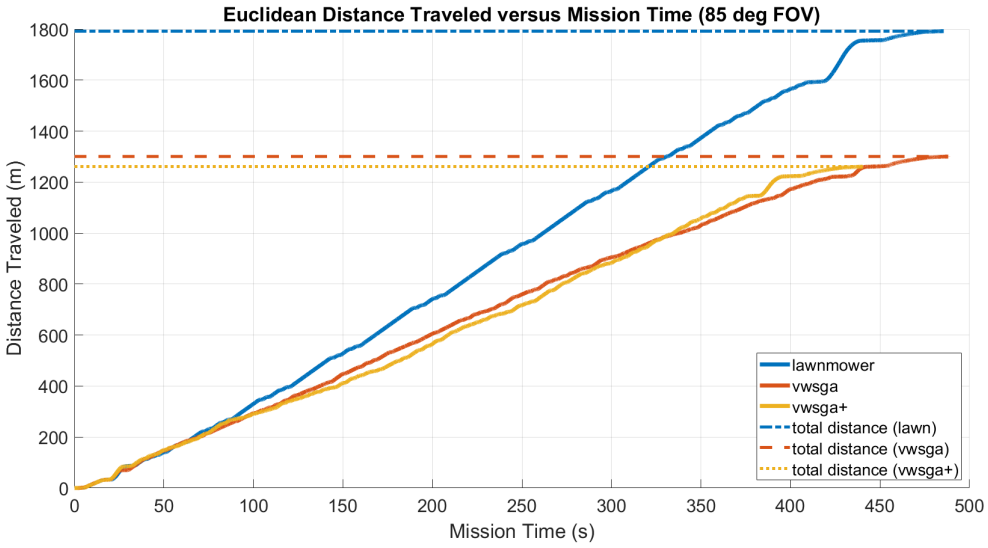


Figure C.2: Flight Test Distance Traveled Comparison for 85° FOV Missions

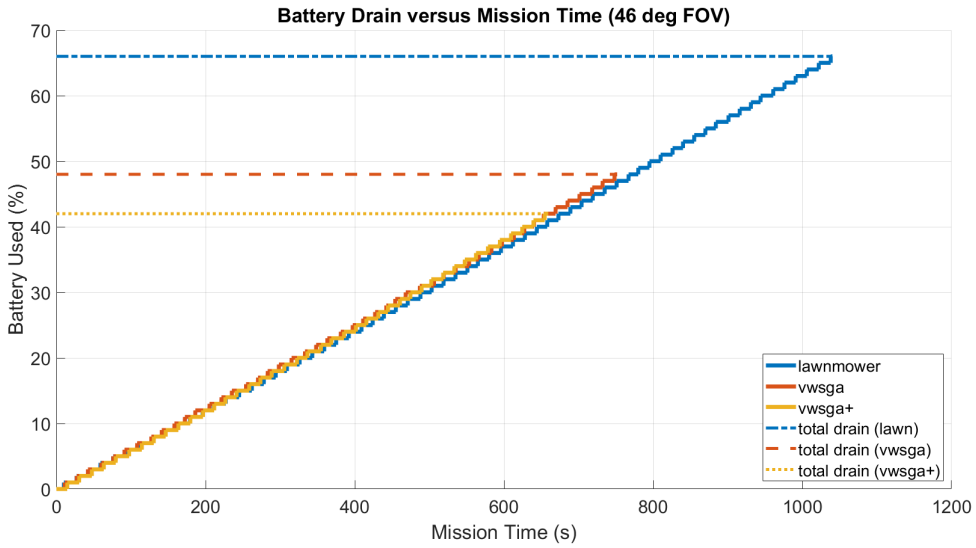


Figure C.3: Flight Test Battery Usage Comparison for 46° FOV Missions

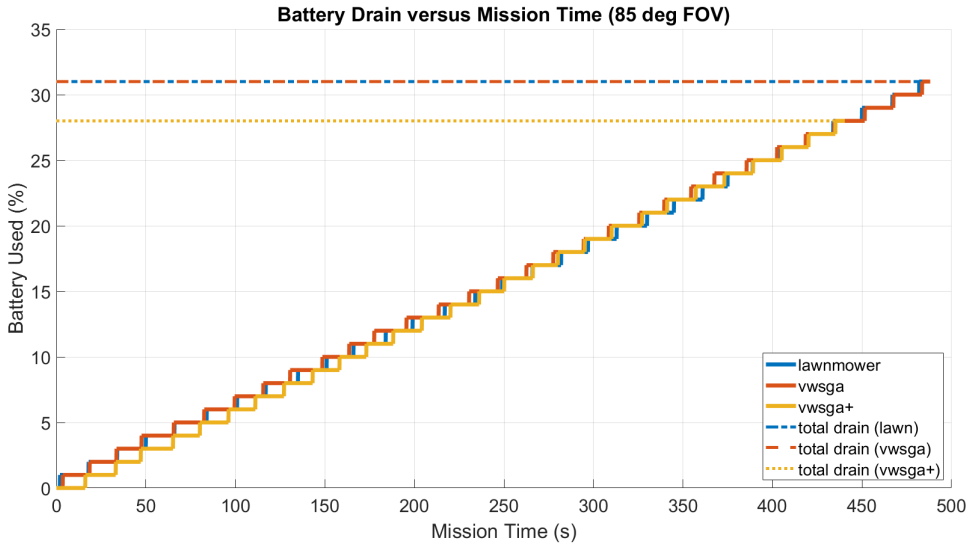


Figure C.4: Flight Test Battery Usage Comparison for 85° FOV Missions

Figure C.5: Cropped Detections (LAWN) for 85° FOV Missions

Figure C.6: Cropped Detections (VWSGA) for 85° FOV Missions

Figure C.7: Cropped Detections (VWSGA+) for 85° FOV Missions

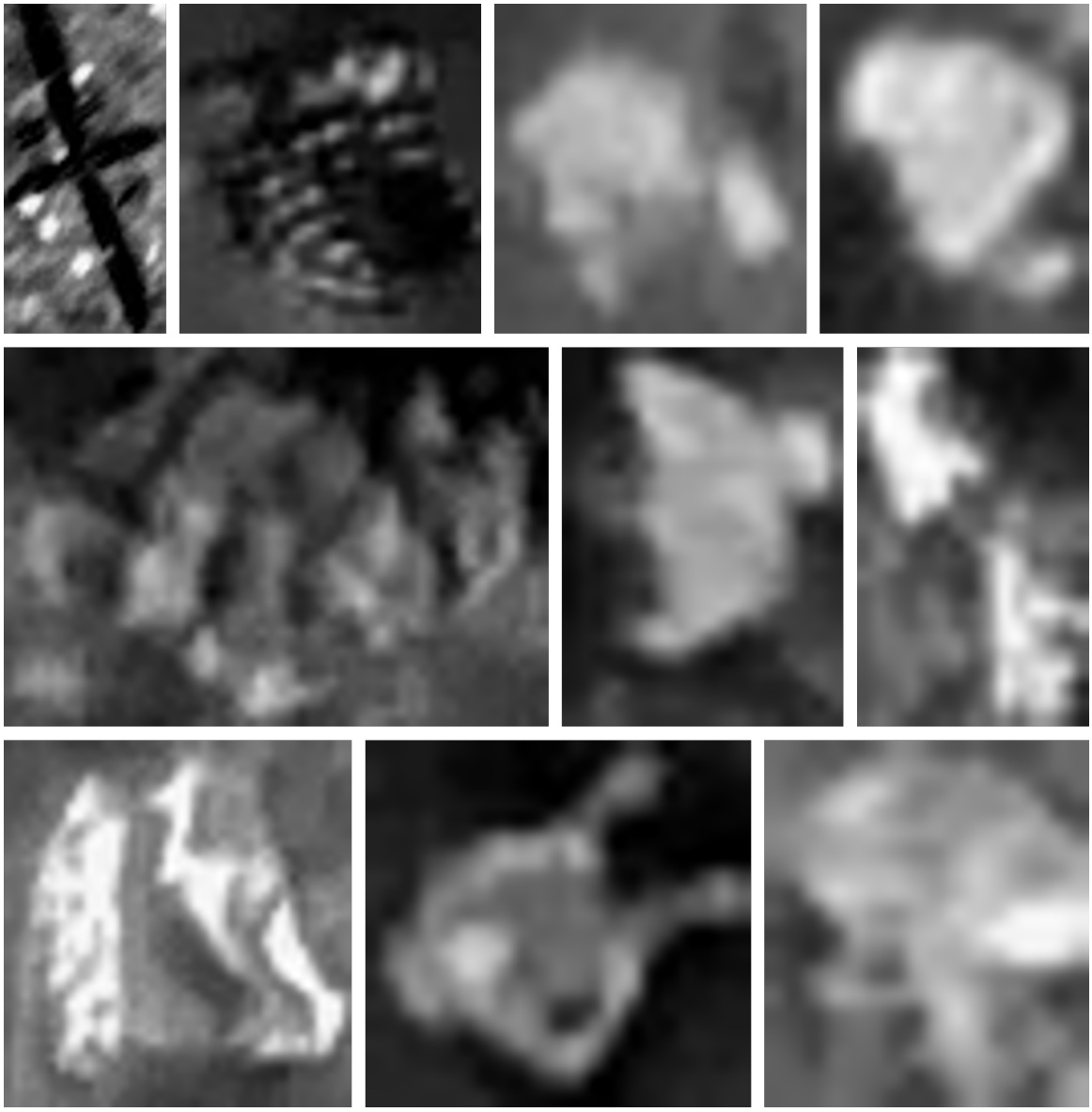


Figure C.8: Cropped Detections (LAWN) for 46° FOV Missions

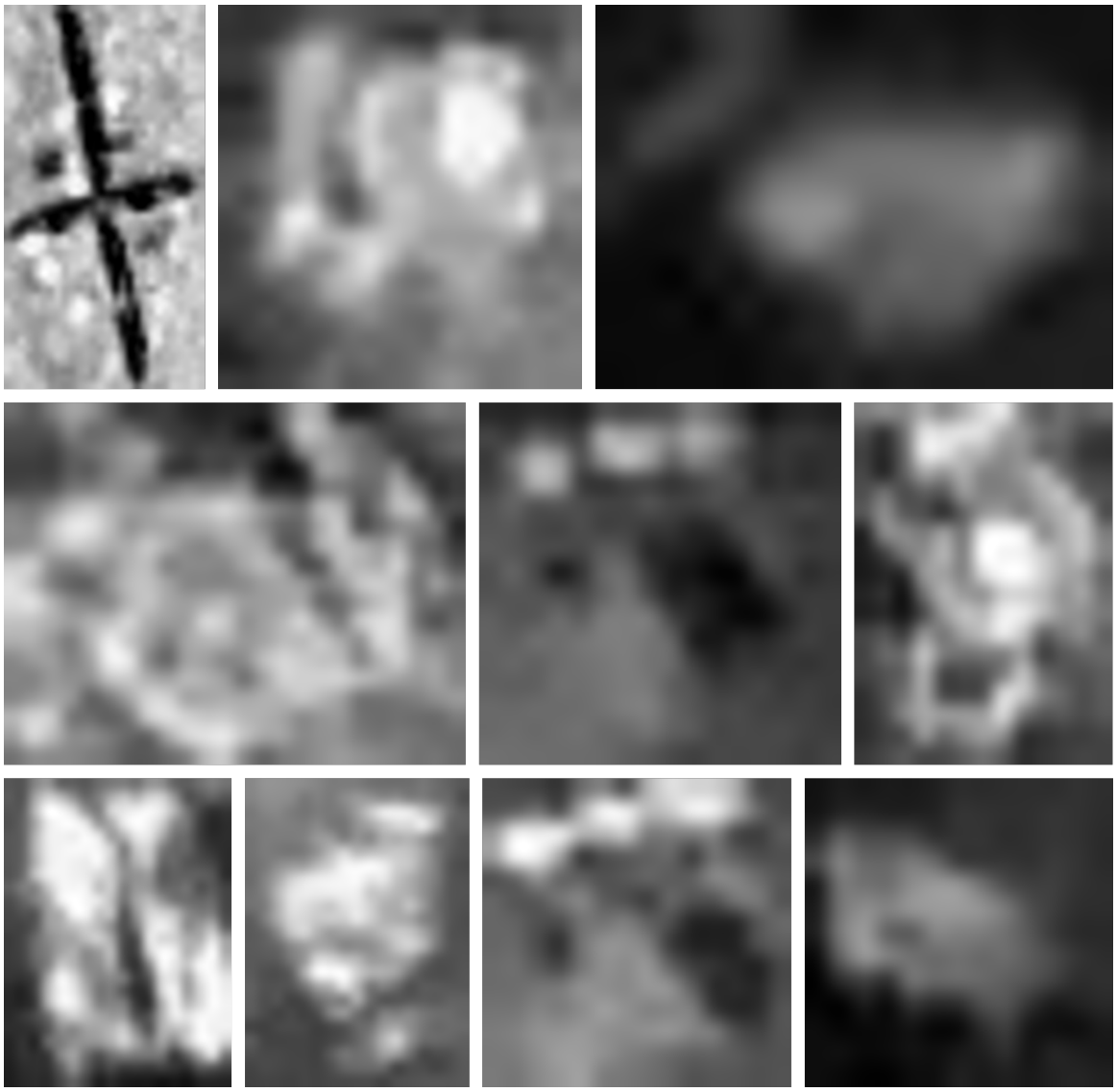


Figure C.9: Cropped Detections (VWSGA) for 46° FOV Missions

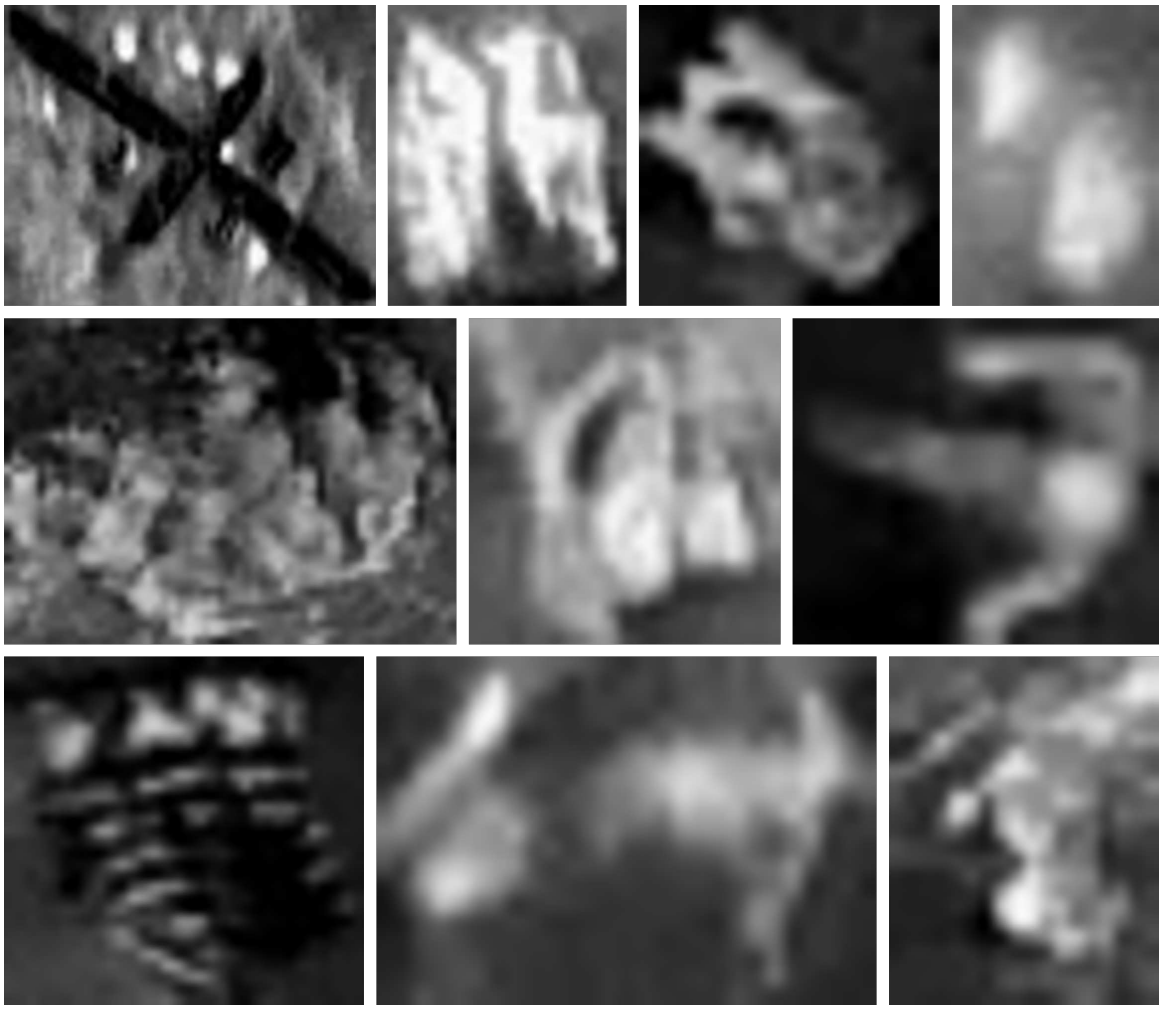


Figure C.10: Cropped Detections (VWSGA+) for 46° FOV Missions