ABSTRACT

Title of Thesis: Vantage Point Path Planning for Efficient UAV Coverage Control

Alec Luterman
Master of Science, 2026

Thesis Directed by: Professor Derek Paley
Department of Computer Science

During mass casualty incidents, unmanned aerial vehicles (UAVs) are a valuable tool to quickly locate casualties in need of triage. Coverage path planning is the task of finding a path plan for a UAV's sensor footprint to cover an entire search domain. Traditional methods, such as lawnmower patterns, take an overly simplistic look at the sensor footprint, resulting in coverage path plans that are not efficient in the amount of time to cover a search domain or efficient in their distribution of coverage. This thesis proposes a coverage path planning algorithm based on generating a set of stationary vantage points that the UAV will travel to and capture imagery. These vantage points ensure that the coverage path plan achieves a spatial resolution threshold throughout the entire search domain, while also minimizing the amount of unnecessary excess coverage both inside and outside of the search domain. Additionally, an integer linear program is formulated to maximize the portion of the search area covered when UAV search time is limited. Through simulation and experimental testing, the proposed coverage path planner is evaluated for its efficiency of coverage and speed in covering a search domain.

Vantage Point Planning For Efficient UAV Coverage Control

by

Alec Michael Luterman

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2026

Advisory Committee:
Professor Derek Paley, Chair/Advisor
Associate Professor Pratap Tokekar
Professor Calin Belta

© Copyright by
Alec Michael Luterman
2026

Acknowledgments

This research was supported by ArtIAMAS Project 1.5, Applied Aerial Autonomy for Long Distance Collaborative Operations, as part of Army Cooperative Agreement W911NF2120076 with DEVCOM Army Research Laboratory (ARL).

I want to start off by thanking my advisor, Dr. Derek Paley, for granting me the opportunity to work on Project 1.5 and for his continuous support throughout my entire time working for him. About three years ago from the time of writing, while still an undergrad, I emailed Dr. Paley about joining the Autonomous Micro Air Vehicle (AMAV) Team. It has been an absolute pleasure working for him these past three years, starting from my time with AMAV and culminating in this thesis. I am truly grateful for the trust that he has placed in me and for his guidance during this entire time.

I would like to also thank the team working on Project 1.5. Starting off, this thesis would not have been possible without the contributions of Zach Bortoff. Throughout this entire project, Zach served as a mentor, constantly providing valuable ideas and insights to this project. In particular, the initial idea and implementation of using the Set Cover Problem to generate vantage points for coverage path planning came from Zach. I would also like to thank Dr. Steve Nogar, our main collaborator from ARL. Anytime I needed help with MAVericks or the Grazers, Steve was always quick to respond and always provided valuable information. Steve's feedback continually pushed my research ideas, in turn leading to some insights like the optimal vantage point altitude. Additionally, thank you to Mike Rawding for his guidance on setting up the Grazers, which I would not have been able to complete without his help.

The team at the University of Maryland UAS Research and Operations Center (UROC) also played a major impact in the success of Project 1.5. Without UROC, none of the flight testing required for the development of this work would have happened. In particular, I would like to thank Chris Titus

for his continuous guidance on everything flight-related, from help setting up the Grazers, insights into UAV flight dynamics, or just simply how to operate UAVs. Alongside Chris, Alex Vrbka, Josh Gaus, Grant Williams, and Rob Neuner all played a crucial role in ensuring safe operations during all of our flight testing, allowing me to focus on the research-side of testing.

Outside of Project 1.5, there were still many people who played a crucial role in the success of this thesis. I would like to first thank Dr. Bruce Golden for his consultation on Vehicle Routing Problems, which in turn helped me understand the nuances of formulating these problems, and in turn helped me land on the Coverage Orienteering Problem. I would also like to thank Dr. Deep Ray for his consultation on efficient ways to structure data for before solving this problem. I would like to thank the entirety of the Collective Dynamics and Control Laboratory for their support throughout my time in the lab. In particular, Ahmed Ashry and Srijal Poojari were incredibly helpful in the process of me writing and defending my thesis. In closing, I would like to give a special thank you to Dr. Pratap Tokekar and Dr. Calin Belta for taking the time to serve on my committee and for their valuable feedback which significantly strengthened my final thesis.

Table of Contents

Acknowledgments ii

Table of Contents iv

List of Tables vi

List of Figures vii

Abbreviations viii

Chapter 1: Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Relation to Previous and Ongoing Work . . . 3
- 1.3 Technical Approach . . . 6
- 1.4 Contributions . . . 8
- 1.5 Outline of Thesis . . . 8

Chapter 2: Background 10
- 2.1 Constraint Optimization Problems . . . 10
  - 2.1.1 Overview . . . 10
  - 2.1.2 Types of Constraint Optimization Problems . . . 11
  - 2.1.3 Google OR-Tools . . . 12
  - 2.1.4 Set Cover Problem . . . 12
- 2.2 Vehicle Routing Problems . . . 13
  - 2.2.1 Traveling Salesman Problem . . . 14
  - 2.2.2 Prize Collecting Traveling Salesman Problem . . . 16
  - 2.2.3 Orienteering Problem . . . 19
  - 2.2.4 Other Relevant Vehicle Routing Problems . . . 21

Chapter 3: Vantage Point Generation 24
- 3.1 Preliminaries . . . 24
  - 3.1.1 Motivation . . . 24
  - 3.1.2 Pinhole Camera Model . . . 26
  - 3.1.3 Spatial Resolution Throughout Sensor Footprint . . . 30
- 3.2 Problem Formulation . . . 34
  - 3.2.1 Vantage Point Definition . . . 34
  - 3.2.2 Problem Statement . . . 35
  - 3.2.3 Integer Linear Program Formulation . . . 36
- 3.3 Solving Methods . . . 38
  - 3.3.1 Setup . . . 38
  - 3.3.2 Vantage Point Selection Methods . . . 41

Chapter 4: Vantage Point Sequencing 43
4.1 Coverage Orienteering Problem . . . 43
4.1.1 Motivation . . . 43
4.1.2 Formulation . . . 45
4.2 Real-Time Routing Algorithms . . . 47
4.2.1 Real-Time Routing . . . 48
4.2.2 Solving Methods . . . 49
4.3 Extension to Dynamic Search . . . 50

Chapter 5: Simulation and Experimental Results 52
5.1 Experimental Testbed . . . 52
5.1.1 MAVericks . . . 52
5.1.2 Hardware . . . 52
5.1.3 Ground Station . . . 53
5.1.4 Testing Location . . . 53
5.1.5 Simulation . . . 54
5.2 Sensor Footprint Analysis . . . 55
5.2.1 Distortion Effects . . . 56
5.2.2 Pixel Density . . . 59
5.3 Simulation Results . . . 62
5.3.1 Methodology . . . 62
5.3.2 Temporal Analysis of Coverage . . . 66
5.3.3 Characterization of Coverage . . . 68
5.3.4 Time-Limited Sequencing . . . 71
5.4 Experimental Results . . . 76

Chapter 6: Conclusion 83
6.1 Summary of Contributions . . . 83
6.2 Suggestions For Future Work . . . 83

Appendix A: Calibration Parameters for Grazer-A Hi-Res Camera 86

Bibliography 87

List of Tables

2.1 Comparison of VRP Variants 23

5.1 Total number of vantage points visited by each routing method within time limit. . . . 75

A.1 Camera Intrinsic and Distortion Parameters 86

List of Figures

1.1 Example lawnmower coverage path plan. Blue dots represent waypoints points the UAV will stop at to take imagery. . . . 2

2.1 Left: An infeasible solution to the TSP with two subtours, violating MTZ constraints. Right: A feasible solution to the TSP with a single tour. . . . 17

3.1 Sensor footprint of Grazer-A hi-res camera at an altitude of 46 meters and a pitch of 60° from nadir to the direction of travel. . . . 34
3.2 Example vantage point and pixel density map using Grazer-A hi-res Camera at an altitude of 46 meters and a pitch of 60° from nadir to the direction of travel. . . . 35

4.1 Framework for solving VRPs for real-time routing of vantage point-based coverage path plans. . . . 49

5.1 ARL Grazer-A [1] . . . 53
5.2 GCS used for flight testing. . . . 54
5.3 Raley Farm test site in southern Maryland. . . . 54
5.4 MAVericks Unity-based simulation environment depicting a Grazer-A UAV flying at the Raley Farm test facility [1]. . . . 55
5.5 Projection of all pixels from the Grazer-A hi-res camera to the ground plane at an angle of 60° from nadir to the direction of travel at an altitude of 46 meters. . . . 56
5.6 Effect of lens distortion on the area of Grazer-A hi-res camera's sensor footprint at different altitudes and camera orientations. . . . 58
5.7 Pixel density within sensor footprint of the Grazer-A hi-res camera. . . . 59
5.8 Spatial resolution within sensor footprint. . . . 60
5.9 Spatial resolution within effective sensor footprint with spatial resolution threshold of 9.5 cm/px. . . . 60
5.10 Difference in area of effective sensor footprint with and without lens distortion. . . . 61
5.11 Comparison of size of effective sensor footprint at different altitudes and camera pitches. 63
5.12 Vantage point coverage path plans flown in real-life flight testing at Raley Farm. Flights were conducted at an altitude of 15 m with a spatial resolution threshold of 10 cm/px. Flight plans differ in the method used for solving Equation (3.12). . . . 65
5.13 Percentage of search domain covered over time for different spatial resolution thresholds and mission types. . . . 67
5.14 Vantage point coverage path plans flown in simulation at a spatial resolution of 5 cm/px and at an altitude of 19.5 m. . . . 69
5.15 Characterization of vantage point coverage. n is the number of cell instances covered by vantage points. . . . 70
5.16 Search domain for time-limited flights in simulation. . . . 72

5.17 Percentage of search domain covered within time limit. 73
5.18 Characterization of coverage for different routing methods and time limits at a spatial resolution of 5 cm/px. 74
5.19 Coverage over time for different routing methods at different time limits and a spatial resolution threshold of 5 cm/px. 75
5.20 Real-world flight testing results with a spatial resolution threshold of 10 cm/px. 78
5.21 Effective sensor footprints at actual obtained vantage points from real-world testing at a spatial resolution of 10 cm/px. 79
5.22 Coverage maps using different levels of vantage point fidelity from real-world flight testing at a spatial resolution of 10 cm/px for the best feasible solution of 2 minutes of solving time from CP-SAT. 81
5.23 Real-world flight testing results with a spatial resolution threshold of 10 cm/px, ignoring UAV pitch and roll. 82

List of Abbreviations

ARL	Army Research Laboratory
CCOP	Clustered Coverage Orienteering Problem
GCS	Ground Control Software
GSD	Ground Sampling Distance
GTSP	Generalized Traveling Salesman Problem
ILP	Integer Linear Program
LNS	Large Neighborhood Search
LP	Linear Program
MILP	Mixed-Integer Linear Program
MTZ	Miller-Tucker-Zemlin
OP	Orienteering Problem
PC-TSP	Prize Collecting Traveling Salesman Problem
QGC	QGroundControl
RC	Remote Control
RGB	Red, Green, Blue
ROS2	Robot Operating System 2
TSP	Traveling Salesman Problem
TSPN	Traveling Salesman Problem with Neighborhoods
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
UMD	University of Maryland
UROC	UMD UAS Research and Operations Center
USV	Unmanned Surface Vehicle
UTM	Universal Transverse Mercator
VRP	Vehicle Routing Problem
WRP	Watchman Route Problem

Chapter 1: Introduction

1.1 Motivation

During mass casualty incidents, the ability to rapidly locate and triage casualties has a strong impact on the number of casualties that survive [2]. If medics have to spend time searching for casualties, then that takes away from the time they can perform triage. Unmanned aerial vehicles (UAVs) can be used in disaster response to free up medics and allow them to focus on triage instead of search since UAVs can locate casualties while covering regions of interest faster than humans can [3,4]. In addition, UAVs are capable of inspecting regions of interest that are dangerous for humans, keeping medics out of harm's way [3].

Coverage path planning is the task of finding a path plan such that the sensor footprint of the UAV covers the entire search domain [5, 6]. Boustrophedon patterns, often called lawnmower patterns, are one of the most common forms of coverage path planning, and are foundational to much of the existing literature [5–7]. In fact, the prevalence of this pattern is reflected by its integration in many UAS mission planners, such as QGroundControl (QGC) [8], which has a built-in tool to design lawnmower coverage path plans and upload to the UAV to execute. Lawnmower coverage path plans are a back and forth pattern of parallel lines, called sweeps or tracks, that reach from one end of the search domain to the other. Once the robot reaches one end of the search domain, it traverses a small distance perpendicular to these tracks before traversing a new parallel track. Figure 1.1 shows an example of a lawnmower coverage path plan, where the longer stretches are the tracks.

The main parameters for this pattern are the flight altitude, track spacing, sweep direction, and the distance between inspection waypoints. Track spacing is the distance between two adjacent parallel tracks and creates a tradeoff between the length of the coverage path plan and the amount of

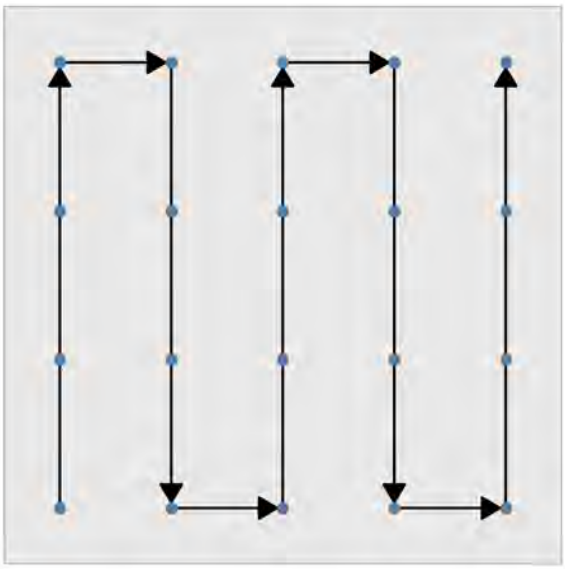


Figure 1.1: Example lawnmower coverage path plan. Blue dots represent waypoints points the UAV will stop at to take imagery.

overlapping coverage. When creating lawnmower mission plans, operators often plan for a defined percentage of overlap between the sensor footprints of adjacent tracks as a way to guarantee complete coverage during flight [7]. This overlap percentage is applied uniformly to the entire coverage path plan and does not consider where additional coverage is actually needed.

In previous work, lawnmower search patterns were shown to result in a very uneven distribution of coverage, with many regions of the search domain being over sampled while others are under sampled [9]. Regions that are over sampled waste precious time collecting unnecessary redundant data while regions that are under sampled might not provide enough coverage to accurately detect targets. Based on this inefficiency, we want a coverage path planning algorithm that results in an even distribution of coverage, while still guaranteeing every portion of the search domain achieves a coverage threshold to ensure we detect and localize every potential target of interest. It is important to note that while there is some benefit to overlapping coverage, in the context of time-constrained

search, overlap should be minimized to cover the search domain as quickly as possible.

While some coverage path plans assume continuous coverage, where the drone does not stop for inspection, in the context of object detection it is better to have defined stops due to the degradation of object detection quality with a moving camera [10]. In addition, being stationary also increases localization accuracy due to minimizing the effects of latency between coordinate frame transformations. This motivates the coverage path plan to be based on a set of vantage points that the UAV will be stationary at to perform inspection, where the cumulative sensor footprint of all of these vantage points must cover the entire search domain.

1.2 Relation to Previous and Ongoing Work

Intrinsic to any coverage path planning algorithm is how the sensor footprint is modeled. Unfortunately, most existing approaches take a simplified view of the sensor footprint.

Starting off on the commercial side, QGC's survey tool designs a lawnmower pattern that can achieve a desired spatial resolution given the UAV's camera parameters and desired percentage of overlap. These are used to determine the maximum altitude the UAV can fly to in order to achieve that spatial resolution, which in turn maximizes the width of the sensor footprint. The track spacing is then defined from the sensor footprint width and overlap percentage. The main issue with this approach is the assumptions made about the camera. In particular, there is no setting for the desired pitch of the camera nor is there a way to account for any lens distortion effects. As a result, the calculated sensor footprint ends up being a rectangle with a uniform spatial resolution throughout. If the UAV camera is oriented with a non-zero pitch, this model does not capture the trapezoidal shape of the sensor footprint and the non-uniform spatial resolution.

To account for non-zero pitch, [11] defines this trapezoidal sensor footprint by its leading edge, trailing edge, center point, and center width. If the effects of lens distortion are ignored, this trapezoid can also be calculated by finding the ground plane intersection point for the four corner pixels in the image. The spatial resolution within this sensor footprint is no longer uniform. One possible solution is to use a representative spatial resolution for the entire sensor footprint. The typical measure of spatial resolution is called ground sampling distance (GSD) which measures the distance between the ground plane intersection points for two consecutive pixel centers in the image [12]. GSD is defined locally for a specific region on the ground as opposed to a global value for the entire sensor footprint. For their representative point, [11] chooses the maximum value between the horizontal GSD and vertical GSD at the center point of the trapezoid, taking a conservative estimate for the spatial resolution. The primary limitation of this approach is that characterizing the entire sensor footprint by a single representative GSD incorrectly treats the quality of coverage within the sensor footprint as uniform. As a result, the coverage path plan will inevitably have under sampled and over sampled regions.

[13] takes a similar approach to this thesis, modeling the generation of vantage points as the 3D Art Gallery Problem, before determining the route between vantage points. The major difference between this thesis and their work is how the sensor footprint is modeled. In [13], the coverage at each vantage point is viewed as a visibility cone, based only on the field of view of the camera, which is pointing at nadir, and the altitude of the vantage point. This model also provides no formal guarantee of a spatial resolution threshold, with resolution addressed only by choosing the minimum possible flight altitude. One other distinction is that this paper is based on utilizing fixed-wing UAVs, which exhibit different flight dynamics than a quadcopter. Most notably, a fixed-wing UAV is not able to stop at a vantage point to perform inspection. [14] also use a similar approach, but focuses only on the deployment of UAVs to stationary vantage points for continuous monitoring.

The goal of the (2D) Art Gallery Problem is to find the minimum number of guards and their placement inside a simple polygon with n walls such that the guards, who can survey 360° around, can collectively see every point in the polygon [15]. The 3D Art Gallery Problem is an extension of the original version and tries to find the minimum number of guards and their placement to cover a 3D terrain [16]. One key limitation is that since guards are assumed to have 360° visibility, there is not a way to directly model a UAV's sensor footprint with lens distortion and pitch. Using a visibility cone like [13, 14] discards the true geometry of the sensor footprint and provides no way to guarantee a spatial resolution threshold is met throughout the search domain.

[17, 18] also model coverage as a visibility cone based on the camera's field of view. However, as opposed to the cone representing the sensor footprint of the UAV at a vantage point, in these works, each cone represents the possible vantage point locations that can see each point on the ground. In particular, [18] also accounts for occlusions and allows for the UAV camera to not be oriented at nadir. To generate vantage points, this work formulates a variant of the Traveling Salesman Problem with Neighborhoods (TSPN). Given a set of neighborhoods, the goal of the TSPN is to find the shortest length tour that visits one point in each neighborhood, thus the shortest length tour that can see every point on the ground.

[19] also models coverage path planning as generating a set and route of vantage points. Their problem formulation is a generalization of the 2D Art Gallery Problem and also the Watchman Route Problem (WRP). The goal of the WRP is to find the minimum length tour for a set of robots to see every point in a polygonal search domain. This problem is modeled as an instance of the Generalized Traveling Salesman Problem (GTSP), which requires the robots to collectively visit one vantage point within each cluster. Note that the main difference between the GTSP and the TSPN is that the GTSP uses discrete points in each cluster while the points in the domain of the TSPN are continuous. Both

the GTSP and the TSPN allow routing and vantage point selection to be modeled as a single problem instead of two separate problems. [20] builds off of this work for an application to infrastructure inspection using a UAV, when the geometric structure of the object of interest may be unknown a priori. This task is also modeled as an instance of the GTSP, ensuring every voxel of the bridge is seen by at least one vantage point. While RGB imagery is used for the actual inspection of the structure, LIDAR is used to build the 3D occupancy model used by the planner to ensure complete inspection of the entire structure.

[21] models the task of 3D object inspection using the Submodular Orienteering Problem. This formulation shares two key similarities with the approach proposed in this thesis. First is the idea that duplicate coverage of a specific region of interest has diminishing value, motivating the definition of a submodular function as the objective function in their problem formulation. Another key similarity is the use of the Orienteering Problem framework to ensure the total coverage time is within a time limit. Though meant for a different task than the scope of this thesis, this work highlights one possible method for optimizing coverage within time limits.

None of the reviewed approaches model the sensor footprint with full fidelity. Consequently, they fail to accurately measure the shape of the sensor footprint and the spatial resolution throughout the sensor footprint. Thus, a different approach has to be taken to guarantee full coverage with a spatial resolution threshold.

1.3 Technical Approach

This thesis presents a new UAV coverage path planning algorithm aimed at maximizing the efficiency of coverage during search and rescue operations. These coverage path plans are based

on generating and sequencing a set of stationary poses called vantage points that the UAV will be stationary at for a short duration in order to capture imagery to perform inspection. In particular, we focus on single-agent coverage path planning with a quadcopter UAV.

Each vantage point is associated with an accompanying sensor footprint and pixel density map, which are generated by calculating the ground plane intersection for every pixel in the image of a camera onboard a UAV using the Pinhole Camera Model. We solve for where to place these vantage points throughout the search domain through the application of a constraint optimization problem called the Set Cover Problem. The solution to this problem minimizes the number of vantage points selected to sufficiently cover the search domain, improving the efficiency of the coverage path plan. In addition, this set of vantage points guarantees that every portion of the search domain achieves a spatial resolution threshold based on the number of ground-plane intersections from pixels in the image of the UAV camera at each vantage point.

When faced with a hard time limit, we formulate an integer linear program (ILP) that selects a subset of vantage points to maximize the area of the search domain that can be covered. The key nuance that this ILP handles is overlapping coverage between vantage points, since with very limited time to search, overlap is not beneficial to the coverage path plan.

Finally, we take a look at how these algorithms actually transfer to practice by first taking a look at how different poses of the camera affect the UAV's sensor footprint. We then show results from experimental and simulation testing to evaluate our coverage path planning algorithm based on the efficiency and quality of coverage.

1.4 Contributions

The primary contributions of this work are as follows:

1. A method for UAV coverage path planning that generates a set of vantage points based on a calculation of the ground plane intersections for all pixels in the camera. The collective coverage of these vantage points ensure every portion of the search domain achieves a desired spatial resolution, while minimizing the amount of overlapping coverage and coverage wasted outside of the search domain.

2. A method of sequencing vantage points to maximize coverage within a time limit. This is a formulation of an integer linear program, that is a variant of the orienteering problem, to maximize the proportion of the search domain achieving a spatial resolution threshold from a vantage point-based coverage path plan within a time limit.

3. Simulation and experimental results to compare coverage efficiency between our coverage path planning algorithm and a lawnmower-derived benchmark.

At the time of writing, many of these contributions have been accepted and are set to appear in publication [22].

1.5 Outline of Thesis

Chapter 2 gives a comprehensive overview of constraint optimization problems, with a particular emphasis on vehicle routing problems and the nuances of their formulations. Chapter 3 explains the relevant camera models for accurately estimating the quality of coverage from a camera onboard

a UAV, leading to the sensor footprint-based coverage path planning algorithm. Chapter 4 details the nuances of routing vantage point-based coverage path plans with regard to temporal constraints. Chapter 5 shows simulation and experimental results evaluating the coverage from vantage points. Chapter 6 summarizes this work and makes suggestions for future work.

Chapter 2: Background

2.1 Constraint Optimization Problems

2.1.1 Overview

Optimization is the task of determining what values of a set of decision variables x should be chosen to maximize or minimize an objective function $f(x)$ [23–25]. If we place restrictions called constraints on the possible values of x then this task becomes known as a constraint optimization problem. These constraints can come in the form of both equality and inequality expressions of x that must be satisfied for any possible solution to the problem. If some x satisfies all constraints, x is referred to as a feasible solution. The optimal solution to a constraint optimization problem is the feasible solution that maximizes or minimizes the objective function. The general form of this problem can be seen in Equation (2.1) below.

$$\begin{aligned}
&\text{Minimize} && f(x) && && \text{(2.1a)}\\
&\text{Subject to} && x \in \mathbb{R}^n && && \text{(2.1b)}\\
& && C_i(x) = 0 && \forall i \in \mathcal{E} && \text{(2.1c)}\\
& && C_i(x) \geq 0 && \forall i \in \mathcal{I} && \text{(2.1d)}
\end{aligned}$$

In this formulation x is the set of decision variables, treated as a vector, and $f(x)$ is the objective function to be minimized, where $f : \mathbb{R}^n \to \mathbb{R}$. The objective function is defined in Equation (2.1a) and the domain of x is defined in Equation (2.1b). Equation (2.1c) defines the set of equality constraints while Equation (2.1d) defines the set of inequality constraints. The sets $\mathcal{E}$ and $\mathcal{I}$ represent the constraint indices, each associated with a constraint function $C_i(x)$ for every $i \in \mathcal{E}$ and $i \in \mathcal{I}$. Since the codomain

of the objective function is a scalar, if we want to maximize some objective function $f(x)$ subject to any set of constraints $C_i(x)$, this is equivalent to minimizing $-f(x)$ subject to the set of constraints $C_i(x)$. Therefore, Equation (2.1) is sufficient to describe constraint optimization problems where the goal is to either maximize or minimize the objective function.

2.1.2 Types of Constraint Optimization Problems

Depending on the classification of the objective function, constraints, and decision variables, we can get different classes of constraint optimization problems. For example, a linear programming problem is a constraint optimization problem where the objective function and all of the constraints are linear. Note that the term programming, not referring to software programming, can often be used interchangeably with the term optimization [23]. A major benefit of using linear programs (LPs) is that any local solution is a global optimal solution. However, LPs with real-valued variables might not be the best choice to model a given problem. For example, if we want to design a constraint optimization problem that chooses waypoints for a UAV to travel to, the choice to include or exclude a given waypoint in the path plan is a binary decision, which should be represented using integer-valued variables. If continuous real-valued variables are used to model this problem as an LP, then there is no guarantee that the optimal solution $x \in \mathbb{R}^n$ to the LP will remain optimal or feasible if x is rounded to be in $\mathbb{Z}^n$ [24]. An integer linear programming (ILP) problem is an LP where all decision variables are constrained to only take integer values. Unfortunately, while LPs can be solved in polynomial time, ILP problems are in NP-Complete [26].

2.1.3 Google OR-Tools

Google OR-Tools is an open source software suite for solving optimization problems, including constraint optimization problems, LPs, and ILPs [27]. One benefit of using OR-Tools is their Routing Library, which is highly optimized for solving vehicle routing problems (VRPs). In addition, their Routing library allows the user to ignore many of the unintuitive constraints typically found in VRPs and instead focus on the high level objectives and constraints of the problem [28]. Another reason to use OR-Tools is that their main constraint optimization problem solver, called CP-SAT, is consistently one of the top performing solvers on benchmark problems [29]. One thing of note is that both the Routing Library and CP-SAT solver only support integer values in order to increase computational speed.

2.1.4 Set Cover Problem

One ILP that is utilized in this work is called the Set Cover Problem which given a universe of elements and a collection of subsets of that universe, finds the minimum cardinality selection of subsets that collectively contain all elements in the universe. [25]. While there exists a weighted variant of this problem, where each subset has a weight and the goal is to minimize the total weight of selected subsets, we focus on the unweighted variant. A formulation of the Set Cover Problem is shown in Equation (2.2) below. Formally, let $U = \{e_i\}_{i=1}^{n}$ be the universe of elements to cover, and let $S = \{s_1, \ldots, s_m\}$ be the set of subsets of the universe where $s_j \subseteq U \; \forall j \in \{1, \ldots, m\}$.

$$\text{Minimize} \quad \sum_{j=1}^{m} x_j \tag{2.2a}$$

$$\text{Subject to} \quad x_j \in \{0,1\} \qquad \forall j \in \{1,\ldots,m\} \tag{2.2b}$$

$$\sum_{j:e_i \in s_j} x_j \geq 1 \qquad \forall i \in \{1,\ldots,n\} \tag{2.2c}$$

In this formulation, for every index j, x_j is a binary variable denoting if subset s_j has been selected to cover U. Equation (2.2b) is the set of constraints forcing x_j to be binary-valued. Since $x_j = 1$ if and only if s_j is selected, the sum of all x_j gives the count of the number of subsets selected. This logic is applied to the objective function in Equation (2.2a) to minimize the number of subsets selected. Equation (2.2c) is the set of constraints ensuring every element in U is covered. For each $i \in \{1,\ldots,n\}$, we count the number of selected subsets s_j where $e_i \in s_j$. Suppose $J \subseteq \{1,\ldots,m\}$ denotes the indices of subsets for a potential solution to Equation (2.2). If there exists some $e \in U$ such that $\forall j \in J$, $e \notin s_j$, then the constraint from Equation (2.2c) will be violated as there is no $j \in \{1,\ldots,m\}$ where $e \in s_j$ such that $x_j = 1$.

2.2 Vehicle Routing Problems

Another constraint optimization problem we explore is known as the vehicle routing problem (VRP), which has the goal of determining a set of routes for a team of salesmen to serve a set of customers located at different cities [30–33]. These routes start and end from specialized locations called depots, of which there can be multiple. Some of the factors that can influence a route are cost, time, distance, vehicle capacity, etc. The breadth and number of VRP variants is so large that

it makes it intractable for a singular formal definition that is abstract enough to handle all possible variants. In this work we are interested in determining the route for a singular salesman — in our context, the route for a single UAV. In this section, we will build up the framework for understanding and formulating VRPs using the TSP, then show how different variations of constraints and objectives can be constructed to handle different applications. Note that the VRP is NP-hard since its most basic special case, the Traveling Salesman Problem (TSP), is also NP-hard [33].

2.2.1 Traveling Salesman Problem

One of the most well-known VRPs is the Traveling Salesman Problem (TSP), which has the goal of finding a tour of cities that visits each city exactly once, while minimizing the total travel cost over all potential routes [34–36]. Depending on the needs of the problem, travel cost could be temporal, monetary, distance-based, or some other type of metric. Formally, solving the TSP involves finding the Hamiltonian Cycle of minimum cost for a set of nodes in a weighted complete undirected graph.

Let $G = (V, E)$ be a complete undirected graph where $V = \{0, \ldots, n\}$ denotes the set of cities that the salesman must travel to, where index 0 is the depot, and E is the set of pairwise edges between the cities. Each edge $e_{i,j}, i \neq j$ between cities i and j has an associated weight $c_{i,j}$, which is the cost of traveling from city i to city j. Using an undirected graph results in symmetric travel times between cities, where the time to travel from city i to city j is the same as the travel time from city j to city i. Formally, for $i \neq j$, $c_{i,j} = c_{j,i}$ and $e_{i,j} = e_{j,i}$. A directed graph can be used to handle asymmetric travel times. Equation (2.3) below shows the formulation of the TSP using Miller-Tucker-Zemlin (MTZ) constraints, where $x_{i,j}$ is a binary variable that denotes if the edge $e_{i,j}$ between cities i and j has been selected for the final route [36, 37]. u_i is a variable used to eliminate subtours, which

are cycles that only visit a proper subset of the cities, ensuring the entire tour is continuous. Figure 2.1 shows the comparison between a route with subtours and a route without subtours. Also, if $u_i < u_j$, for $i \neq j$, then city i is visited before city j.

$$\text{Minimize} \quad \sum_{i=0}^{n} \sum_{j=0, i \neq j}^{n} c_{i,j} x_{i,j} \tag{2.3a}$$

$$\text{Subject to} \quad x_{i,j} \in \{0, 1\} \qquad \forall i \neq j \in \{0, \ldots, n\} \tag{2.3b}$$

$$\sum_{i=0, i \neq j}^{n} x_{i,j} = 1 \qquad \forall j \in \{0, \ldots, n\} \tag{2.3c}$$

$$\sum_{j=0, j \neq i}^{n} x_{i,j} = 1 \qquad \forall i \in \{0, \ldots, n\} \tag{2.3d}$$

$$u_i - u_j + 1 \leq n(1 - x_{i,j}) \qquad \forall i \neq j \in \{1, \ldots, n\} \tag{2.3e}$$

$$1 \leq u_i \leq n \qquad \forall i \in \{1, \ldots, n\} \tag{2.3f}$$

In this formulation, the objective function, shown in Equation (2.3a), minimizes the sum of costs over all of the edges selected. By the construction of the constraints in this formulation, $x_{i,j}$ has a value of 1 if the edge $e_{i,j}$ is selected in the final tour, and 0 otherwise. Therefore, we know the objective function only sums the costs of selected edges. The constraint that forces $x_{i,j}$ to be binary-valued is Equation (2.3b). Equation (2.3c) is the constraint that forces the number of inbound edges to each city j to be exactly 1. In layman's terms, this means that we only travel to each city once. Note that starting at the depot is not the same as traveling to the depot - the only time we travel to the depot is at the end of the tour. Since $x_{i,j}$ denotes if the edge $e_{i,j}$ is selected, if j is held constant, then all of the $x_{i,j}$ are all the possible edges that we can take to travel to city j. Therefore, if we only want to travel to city j

exactly once, then only one $x_{i,j}$ over all $i \neq j$ can have a value of 1. Similarly, Equation (2.3d) is the constraint that the number of outbound edges to each city i must be exactly 1. I.e., we only leave each city once.

The constraints related to subtour elimination are Equations (2.3e) and (2.3f), where Equation (2.3f) is the domain of the MTZ variables u. Suppose the edge from city i to city j is selected for a possible tour, where $i \neq j$. Then $x_{i,j} = 1$ and Equation (2.3e) gives $u_i - u_j + 1 \leq n\,(1-1) = 0$. Rearranging, we get $u_j \geq u_i + 1$, which with the interpretation that these variables show the relative ordering of the cities visited, means that city j is visited after city i. Now suppose that the edge from city i to city j was not selected. This gives $u_i - u_j + 1 \leq n\,(1-0)$, which is always true given the bounds on each u. Assume for contradiction, we have cities (a, b, c, d) with depot X, and the solution to the TSP contains the two disconnected subtours $X \to d \to X$ and $a \to b \to c \to a$. Note that this solution satisfies the constraints in Equations (2.3b), (2.3c), and (2.3d). However, based on Equation (2.3e), we have $u_a < u_b < u_c < u_a$, which is a contradiction. Thus, it is not possible to have subtours with this constraint. Figure 2.1 shows how this proposed solution is infeasible, and compares it to a feasible solution to the TSP. Note that since this figure does not include any distances between cities, it is not possible to know if any given feasible solution is also an optimal solution.

2.2.2 Prize Collecting Traveling Salesman Problem

What if it is not in our best interest to travel to every single city? For example, what if the monetary cost to travel to a distant city outweighs the potential benefit of visiting that city? The nominal TSP is not able to handle this variant as it has no way of measuring the value of a city, which leads to the Prize-Collecting TSP (PC-TSP). In the PC-TSP variant, every city now has a prize for

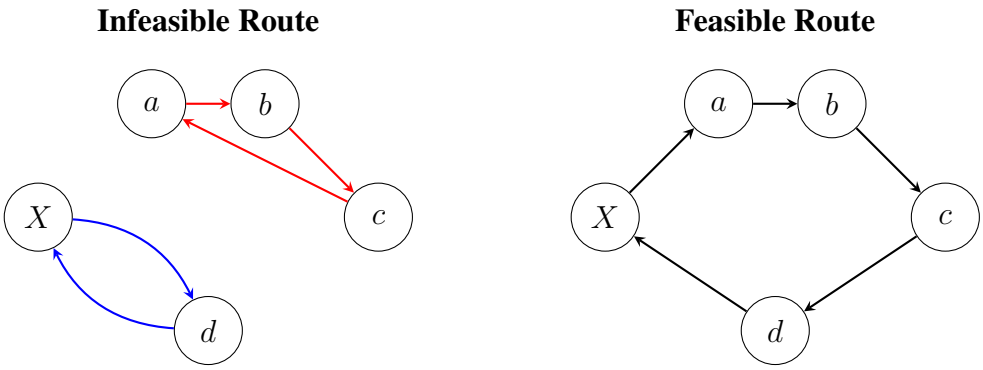


Figure 2.1: Left: An infeasible solution to the TSP with two subtours, violating MTZ constraints. Right: A feasible solution to the TSP with a single tour.

being visited and a penalty if it is not visited. The goal is to find a route and selection of cities such that the sum of travel costs and penalties for cities skipped is minimized while achieving a minimum reward threshold [34, 38].

Formally, the salesman receives a prize w_i if he decides to visit city i, but pays a penalty d_i if he decides to not visit that city. W_0 is the minimum reward threshold that the salesman must achieve. We also define y_i to be the binary variable denoting if city i is included in the tour. All other variables are carried over from the standard TSP formulation. Equation (2.4) below shows the formulation of the PC-TSP.

$$\text{Minimize} \quad \sum_{i=0}^{n} \sum_{j=0, i \neq j}^{n} c_{i,j} x_{i,j} + \sum_{i=1}^{n} d_i \left(1 - y_i\right) \tag{2.4a}$$

$$\text{Subject to} \quad x_{i,j} \in \{0, 1\} \qquad \forall i \neq j \in \{0, \ldots, n\} \tag{2.4b}$$

$$y_i \in \{0, 1\} \qquad \forall i \in \{0, \ldots, n\} \tag{2.4c}$$

$$y_0 = \sum_{j=1}^{n} x_{0,j} = \sum_{i=1}^{n} x_{i,0} = 1 \tag{2.4d}$$

$$y_k = \sum_{i=0, i \neq k}^{n} x_{i,k} = \sum_{j=0, j \neq k}^{n} x_{k,j} \qquad \forall k \in \{1, \ldots, n\} \tag{2.4e}$$

$$\sum_{i=1}^{n} y_i w_i \geq W_0 \tag{2.4f}$$

$$u_i - u_j + 1 \leq n \left(1 - x_{i,j}\right) \qquad \forall i \neq j \in \{1, \ldots, n\} \tag{2.4g}$$

$$1 \leq u_j \leq n \qquad \forall j \in \{1, \ldots, n\} \tag{2.4h}$$

In the objective function of this formulation, shown in Equation (2.4a), there is now a term used to penalize unvisited cities. To constrain the variable representing if a city has been visited to a binary variable, we use Equation (2.4c). Note that based on the depot constraints in Equation (2.4d), y_0 is constrained to always be 1 since we will always visit the depot. The constraints for the number of inbound and outbound edges to each city have now been modified to handle the fact that a city might not be visited. If city k is skipped, then that means there are no inbound edges to it (the salesman does not travel to it) and there are no outbound edges from it. Examining Equation (2.4e), this means over all $i \neq k \in \{0, \ldots, n\}$ all $x_{i,k} = 0$ and over all $j \neq k \in \{0, \ldots, n\}$ all $x_{k,j} = 0$. This is consistent with $y_k = 0$ meaning that city k is skipped. Similar logic follows to show that this constraint enforces the correct behavior if a city is visited. Equation (2.4f) is used to enforce the constraint that the reward

for the cities that the salesman travels to satisfies the minimum reward threshold W_0.

We will show that the MTZ constraints in Equation (2.4g) are able to handle the relaxation of the requirement for the salesman to visit every city. Suppose that both city i and city j are skipped by the salesman. Using Equation (2.4g) we have $u_i - u_j + 1 \leq n\,(1 - x_{i,j})$ which since $x_{i,j} = 0$ results in $u_i - u_j + 1 \leq n$, which is trivially true based on the bounds of each u constrained in Equation (2.4h) as the maximum value of $u_i - u_j$ is $n - 1$. Now, suppose city i is visited but city j is skipped. This also results in $x_{i,j} = 0$, causing Equation (2.4g) to be trivially true like the previous case. If city i is skipped but city j is visited then similar logic applies. If city i and city j are both visited, the MTZ constraints behave the same as before in the nominal TSP.

2.2.3 Orienteering Problem

What if the salesman only has limited travel time? What is the tour of cities that can maximize his profits? These questions lead to the Orienteering Problem (OP). In this problem, the salesman has a maximum travel time limit and each city has a non-negative reward for visiting. The objective is to find the route and selection of cities to maximize the reward gained [34, 39–41]. Note that not every city has to be selected in the final route. This problem is named after one of the variants of the sport orienteering, called the score orienteering event, where the goal is for competitors to visit a set of control points, each with a reward, to maximize their score within a time limit.

Formally, let $T_{\max}$ be the maximum travel time allotted to the salesman and let w_i be the reward associated with city i. Additionally, let $t_{i,j}$ denote the travel time between city i and city j for all cities $i \neq j$. Equation (2.5) below shows the formulation of the OP.

$$\text{Maximize} \quad \sum_{i=1}^{n} w_i y_i \tag{2.5a}$$

$$\text{Subject to} \quad x_{i,j} \in \{0,1\} \qquad \forall i \neq j \in \{0,\ldots,n\} \tag{2.5b}$$

$$y_i \in \{0,1\} \qquad \forall i \in \{0,\ldots,n\} \tag{2.5c}$$

$$y_0 = \sum_{i=1}^{n} x_{i,0} = \sum_{j=1}^{n} x_{0,j} = 1 \tag{2.5d}$$

$$y_k = \sum_{i=0, i\neq k}^{n} x_{i,k} = \sum_{j=0, j\neq k}^{n} x_{k,j} \qquad \forall k \in \{1,\ldots,n\} \tag{2.5e}$$

$$\sum_{i=0}^{n} \sum_{j=0, i\neq j}^{n} x_{i,j} t_{i,j} \leq T_{\max} \tag{2.5f}$$

$$u_i - u_j + 1 \leq n\,(1 - x_{i,j}) \qquad \forall i \neq j \in \{1,\ldots,n\} \tag{2.5g}$$

$$1 \leq u_i \leq n \qquad \forall i \in \{1,\ldots,n\} \tag{2.5h}$$

The objective function in this formulation, shown in Equation (2.5a), is the sum of the rewards for all cities visited. The only constraint that differs from the PC-TSP formulation in Equation (2.4) is the change of constraint for the salesman achieving the minimum reward threshold in Equation (2.4f), to a constraint on the maximum travel time that the salesman can travel.

The OP and PC-TSP are very similar formulations as both try to balance the cost of travel with the reward for cities visited and also support the salesman deciding not to visit every city. However, the major differences between the OP and the PC-TSP are that the OP has cost as a constraint, in the form of travel time, while minimizing the cost including penalty for unvisited cities is the objective of the PC-TSP. Similarly, the PC-TSP has a constraint on the minimum reward the salesman must achieve but the objective of the OP is to maximize the total reward from the cities the salesman visits.

In addition, the OP does not have a mechanism for penalizing skipped cities like the PC-TSP has. The OP is used to maximize a reward value with respect to a cost constraint while the PC-TSP is used to minimize cost with respect to a reward threshold.

2.2.4 Other Relevant Vehicle Routing Problems

One issue with the PC-TSP and the OP is that the reward structure for visiting a city is fixed temporally. That is, the reward for visiting city i is the same if it is the first city the salesman visits or if it is the last city visited. This leads to the Discounted Reward TSP which adds an incentive for each city to be visited as early as possible by introducing an exponential reward decay for each city based on the time the city is visited [42]. Formally, given a decay constant $\gamma \in (0, 1)$, the objective is to maximize

$$\sum_{i=1}^{n} c_i \gamma^{T_i} \tag{2.6}$$

where T_i is the time that city i is visited. The Discounted Reward TSP could possibly be used to incentivize traveling to the most important cities first and create a plan to visit all cities, even if it is uncertain if the salesman will actually be able to visit them.

If we want to keep the reward of visiting a city dependent on the cities visited before it, but use a linear problem instead of the Discounted Reward TSP, we can use the Target Visitation Problem [43]. In this problem, we have a pairwise ordering of rewards where the salesman receives a reward $w_{i,j}$ if city i is visited before city j, for all pairs i, j. Note that this reward is not based on the arrival of the salesman to city i but instead due to the pairwise ordering of visitations. Another VRP variant that incentivizes traveling to the most important cities first is the Hierarchical TSP [44]. In this problem, each city is assigned a priority level and the salesman has to visit all cities of priority class k before

proceeding to any city of priority class $k + 1$.

The closest VRP variant to our work is the Clustered Coverage Orienteering Problem (CCOP) [45], which is a specialized problem to the task of water sampling with unmanned surface vehicles (USV). The goal is to find the selection of water sampling sites for a USV to visit, with constraints on the maximum mission time, the maximum volume of water samples to carry, and the requirement to visit every single cluster of sampling sites and visit a minimum number of sampling sites within each. The aspect that is most relevant to our work is the idea that overlapping coverage provides no additional information, so the objective function should only count unique coverage. Their objective function is to maximize the amount of unique area that is sampled by the USV. The primary distinction between the CCOP and our work is in application.

Table 2.1 gives a summary of all of the VRP variants mentioned in this chapter. This table shows the objective of each variant, the cost metrics used, the reward structure, and if it required for all cities or nodes to be visited.

Table 2.1: Comparison of VRP Variants

Problem	**Objective**	**Cost**	**Reward**	**Visit All?**
TSP	Minimize total travel distance	Distance	None	Yes
PC-TSP	Minimize travel distance plus penalties for unvisited waypoints, subject to a minimum profit threshold	Distance, Unvisited penalties	Fixed, Independent	No
OP	Maximize reward within a time limit	Time	Fixed, Independent	No
Discounted-Reward TSP	Maximize time-decaying reward	Distance	Time-decaying	Yes
Target Visitation Problem	Maximize reward minus costs	Distance	Fixed pairwise relative ordering	Yes
Hierarchical TSP	Minimize distance while respecting priority-level ordering of waypoints	Distance within each priority level	None	Yes
CCOP	Maximize coverage within time limit, visiting at least one node from each cluster, subject to carrying capacity	Time, Water volume	Unique coverage	No

Chapter 3: Vantage Point Generation

This chapter outlines the first portion of a new coverage path planning algorithm based on generating a set of waypoints, called vantage points, that ensure the UAV is able to capture imagery of the entire search domain with a guarantee on achieving a spatial resolution threshold throughout the entire search domain. Section 3.1 outlines the motivation for the structure of our algorithm and our measure of spatial resolution. Given the goals for our coverage path planner in Section 3.1, we give the formal problem statement and the formulation of the ILP for generating vantage points in Section 3.2. Finally, Section 3.3 explains how to solve this ILP to generate the set of vantage points.

3.1 Preliminaries

3.1.1 Motivation

Object detection is the task of identifying and localizing every instance of objects from a class of interest within an input signal, such as an image frame [46]. Note that while the modality of interest for this work is RGB imagery, this task can also be performed using modalities such as radar or sonar as the input data [47, 48]. One method for RGB-based object detection is using a convolutional neural network such as YOLO [49], which outputs a bounding box in the image frame for each object of interest detected. One particular benefit of YOLO is that its single-stage architecture makes it well-suited for real-time inference.

Given a set of targets in the search domain, we want a coverage path plan that maximizes the probability that we correctly detect all of the targets using imagery provided by the UAV. Boustrophedon coverage path plans, commonly referred to as lawnmower patterns, are the conventional method to

perform coverage path planning. With a simple search domain, i.e., a search domain that is a convex polygon, this is just a back-and-forth pattern of parallel lines with a defined spacing between them. Even in the most simple case of a square search domain, this type of coverage path plan results in a very uneven distribution of coverage [9], based on examining the probability of detecting objects of interest throughout the search domain using YOLO.

To estimate this probability of detection for a given coverage path plan, the UAV's poses over time are recorded and used to calculate the actual coverage obtained. Note that the search domain is decomposed into a grid of $1\,\text{m} \times 1\,\text{m}$ cells. At each pose, an instantaneous probability of detection is calculated using only the current coverage, calculating how many pixels from the camera onboard the UAV intersect the ground plane within each cell of the search domain to get a pixels-on-target-derived probability of detection.

Pixels-on-target is a measure of how many pixels from a camera cover an object of interest in the image. Objects that appear smaller in an image, i.e., objects that are represented by fewer pixels in an image, are more difficult to detect than larger objects [50]. Given a trained object detection model and a specific class of object, we can model the probability of detecting a target given a target exists within each cell of the search domain based on the number of pixels-on-target for a singular camera snapshot [9]. Because of the relationship between the number of pixels-on-target and the probability of detection, we want a coverage path plan that guarantees we achieve a minimum quality of coverage everywhere in the search domain. Additionally, we want a coverage path plan that avoids the uneven distribution of coverage and avoids oversampling whenever possible.

Localization is the task of estimating the location of an object in the world frame. In the context of target detection, this is estimating where a detected object is in the world frame after detecting it in an RGB image. This estimation relies on having an accurate estimation of the camera's location,

which in the context of the work requires having an accurate estimation of the UAV's pose at the time imagery was taken. One issue that arises is the fact that a UAV is a moving platform which can cause errors in localization estimation due to the latency of the drone's location, giving an inaccurate estimation of the location of detected targets. This is also coupled with the fact that object detection quality degrades when using a moving camera [10]. This motivates the requirement of the UAV to be stationary in order to perform inspection at any given pose.

3.1.2 Pinhole Camera Model

The pinhole camera model is a mathematical model that represents the projection of the three-dimensional world onto a two-dimensional image using an ideal pinhole camera [51–53]. An ideal pinhole camera describes the camera aperture as a singular point, called the center of projection, and simplifies the model for perspective projection by ignoring lens effects such as distortion or vignetting. The center of projection is the point in space where all of the light rays that are sampled by a pinhole camera intersect. For any 3D point in the world that is depicted in an image, a ray can be constructed that passes through that point and the center of projection whose intersection with the image plane is the point on the image where that 3D point is represented. In a physical pinhole camera, the image plane is the photosensitive surface (sensor) that the image is captured on.

We proceed by defining the coordinate frames used for transforming world points to and from points in the image. The image frame is a Euclidean space in $\mathbb{R}^2$ coincident with the image plane, while the world frame is a Euclidean spaces in $\mathbb{R}^3$ representing points in the world. The scale of the image frame is relative to real-world physical units and its origin is the intersection between the optical axis and the image plane, called the principal point. The optical axis is defined to be the line through

the center of projection orthogonal to the image plane. We then define the pixel frame whose origin is the upper-left corner of the image sensor. While the scale of the pixel frame is defined relative to discrete pixels in the image, which are defined at integer coordinates within the bounds of the image, it is a continuous space allowing for subpixel calculations. Moving to three-dimensional spaces, the camera frame is defined to have its origin at the center of projection with its XY-plane parallel to the image plane and its Z-axis coincident with the optical axis. Typically, we take the world frame to be a coordinate system such as the Universal Transverse Mercator (UTM) system, which is used to transform GPS coordinates from degrees of longitude and latitude into meters. Coordinates in the camera frame denote the location of 3D points in the world relative to the location of the camera.

Suppose we have a point in the world frame $p_w = (x_w, y_w, z_w, 1)^T \in \mathbb{P}^3$ that we want to calculate which pixel in the image it maps to. $\mathbb{P}^3$ is the three-dimensional space of homogeneous coordinates, which allow for translations via matrix-multiplication, since translation cannot be applied to coordinates in $\mathbb{R}^3$ using matrix-multiplication. The homogeneous coordinate system $\mathbb{P}^2$ is also used for the same reason as opposed to $\mathbb{R}^2$. A coordinate $(x_1, \ldots, x_n)^T \in \mathbb{R}^n$ can be mapped to a coordinate in $\mathbb{P}^n$ as $(x_1, \ldots, x_n, 1)^T$, while a coordinate $(x_1, \ldots, x_n, w)^T \in \mathbb{P}^n$ can be mapped to a coordinate in $\mathbb{R}^n$ as $(\frac{x_1}{w}, \ldots, \frac{x_n}{w})^T$, for $w \neq 0$. We use the camera's extrinsic parameters to transform p_w into a point in the camera frame p_c. The extrinsic matrix ${}^cT^w$ defines the transformation from the world frame to the camera frame:

$$ {}^cT^w = \begin{bmatrix} {}^cR^w & -{}^cR^w\tilde{C} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{3.1} $$

Here, $\tilde{C}$ represents the position of the camera's center of projection in the world frame as a vector in $\mathbb{R}^3$ and ${}^cR^w$ represents the camera's orientation as a rotation matrix in $\mathbb{R}^{3\times3}$. We then have $p_c = {}^cT^w p_w$, where $p_c = (x_c, y_c, z_c, 1)^T \in \mathbb{P}^3$ represents the position of p_w in the camera frame. We can calculate

the projection of p_c onto the image frame as

$$(x_i, y_i) = (f\frac{x_c}{z_c}, f\frac{y_c}{z_c}),$$

where f is the distance between the center of projection and the image plane called the focal length.

In practice, we map directly from 3D homogeneous coordinates to coordinates in the pixel frame by using a matrix K, called the calibration matrix, that represents the camera's intrinsic parameters:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

(c_x, c_y) denotes the location in the pixel frame of the principal point. f_x and f_y represent the focal length of the camera in terms of pixel dimensions in each direction. Given the camera's column-wise and row-wise density of pixels in pixels per millimeter m_x and m_y, we calculate $f_x = m_x f$ and $f_y = m_y f$.

Putting this all together, we can calculate the location of the pixel frame $p_{px} = [u, v, w]^T \in \mathbb{P}^2$ that p_w maps to via:

$$p_{px} = K[I_3 \mid \mathbf{0}] \ ({}^cT^w p_w) \tag{3.3}$$

Since $p_{px} \in \mathbb{P}^2$, the location in the pixel frame is $(\frac{u}{w}, \frac{v}{w})$, which is then binned to the closest pixel.

To improve the accuracy of the mapping between points in the world frame and their corresponding image pixels, we need to compensate for the effects of lens distortion. In particular, we utilize the Brown-Conrady distortion model, which is also known as the Plumb Bob distortion model [54]. The

Plumb Bob distortion model is used to account for both radial and tangential lens distortion effects. Radial distortion is caused by the curvature of the camera lens and causes straight lines to appear curved while tangential distortion is from lens decentering and causes the image to be tilted and stretched [53, 55].

Given distortion coefficients $[k_1, k_2, p_1, p_2, k_3]$ and some point in the camera frame $p_c \in \mathbb{P}^3$, where $p_c = (p_x, p_y, p_z, 1)^T$, we can calculate the point in the pixel frame p_{px} corresponding to p_c. Note that k_1, k_2, k_3 are the radial distortion coefficients while p_1, p_2 are the tangential distortion coefficients. First, we compute the projection of p_c onto the normalized image plane as $\tilde{p} \in \mathbb{R}^2$ by

$$\tilde{p} = \left(\frac{p_x}{p_z}, \frac{p_y}{p_z}\right)^T = (r_x, r_y)^T \tag{3.4}$$

Using the distortion model, we then calculate the location of the distorted point $x_d \in \mathbb{R}^2$ in the image plane by

$$x_d = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6)\tilde{p} + \begin{bmatrix} 2p_1 r_x r_y + p_2(r^2 + 2r_x^2) \\ p_1(r^2 + 2r_y^2) + 2p_2 r_x r_y \end{bmatrix} \tag{3.5}$$

where $r = \|\tilde{p}\|_2$. We then express $x_d \in \mathbb{R}^2$ in homogeneous coordinates as $x_d' \in \mathbb{P}^2$ to calculate its corresponding point $p_{px} \in \mathbb{P}^2$ in the pixel frame by

$$p_{px} = K x_d' \tag{3.6}$$

To get the actual discrete pixel location, we convert p_{px} to its non-homogeneous coordinate in $\mathbb{R}^2$, then bin to the closest pixel.

3.1.3 Spatial Resolution Throughout Sensor Footprint

The pinhole camera model gives a framework to determine the pixel in an image that any given point in the world frame maps to. We are interested in the inverse problem: given a pixel in the image, can we determine the point in the world frame that it corresponds to? In particular, we are interested in calculating the sensor footprint, which is the region on the ground that a sensor at a fixed location covers [9]. For an RGB camera, this is the region on the ground taken by the point of intersection with the ground plane for all of the pixels in the image.

We begin with a pixel in the pixel frame $p_{px} \in \mathbb{P}^2$ where $p_{px} = (u, v, 1)^T$. Using the camera calibration matrix K, we compute its inverse K^{-1} which is guaranteed to exist since by construction K is an upper triangular matrix with strictly positive diagonal entries. We then calculate the distorted point in the image frame $x'_d \in \mathbb{P}^2$ corresponding to p_{px} by

$$x'_d = K^{-1} p_{px} \tag{3.7}$$

The next step is to convert x'_d back to its corresponding undistorted point in the normalized image frame $\tilde{p}$. I.e., we want to undistort x'_d. Unfortunately, there is no analytic solution to undistort points that were distorted using Equation (3.5) [56]. One possible remedy is to use an iterative numerical algorithm to solve for the inverse, which is what OpenCV uses for their functions to undistort pixels to normalized image coordinates [54]. Specifically, they use an iterative fixed point algorithm in their function `undistort_points` which works on a sparse set of pixels. In practice, we simply use functions from OpenCV to convert pixels to normalized image coordinates.

Given a normalized image coordinate $\tilde{p} = (p_x, p_y, p_w)^T$ in $\mathbb{P}^2$, we then compute the ray $\mathbf{R_{cam}}$ in

the camera frame that passes through $\tilde{p}$ and the center of projection. Since the center of projection is the origin of the camera frame, the direction of $\mathbf{R_{cam}}$ can be represented by the vector $(p_x/p_w, p_y/p_w, 1)^T$. We want to calculate the parameterized ray $\mathbf{R_{world}}(t)$, for $0 \leq t \in \mathbb{R}$, which represents $\mathbf{R_{cam}}$ in the world frame and gives the mechanics to calculate this ray's points of intersection in the world frame. Following [9], we calculate $\mathbf{R_{world}}(t)$ by

$$\mathbf{R_{world}}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \left(({}^cR^w)^T \mathbf{R_{cam}} \right) t + \tilde{C} \tag{3.8}$$

Note that the inverse of a rotation matrix is its transpose, so if ${}^cR^w$ rotates a point in the world frame to the camera frame, then $({}^cR^w)^T$ rotates a point in the camera frame to the world frame.

Given $\mathbf{R_{world}}(t)$, we now proceed with finding the point of intersection with the ground plane, assuming a flat Earth. This assumption models the world with a constant terrain elevation and ignores the curvature of the Earth, which is negligible at low altitudes. Finding the point of intersection with the ground plane is equivalent to finding the position on the ray where $z(t) = 0$. Let $\tilde{C} = \left(\tilde{C}_x, \tilde{C}_y, \tilde{C}_z\right)^T$ and let $({}^cR^w)^T \mathbf{R_{cam}} = (v_x, v_y, v_z)^T$, then $z(t) = tv_z + \tilde{C}_z$. Let t_0 satisfy $z(t) = 0$, which has a unique solution since $\mathbf{R_{world}}(t)$ is linear, assuming $\mathbf{R_{world}}(t)$ is not parallel to the ground plane. We can then derive

$$t_0 = \frac{-\tilde{C}_z}{v_z} \tag{3.9}$$

Thus, the pixel in the image p_{px} receives its intensity value from the point on the ground plane $(x(t_0), y(t_0), z(t_0))^T \in \mathbb{R}^3$. Note that due to the size of each unit of an image sensor grid, the intensity of each pixel in the image actually corresponds to a small region around the point on the ground plane

that we calculated. However, this fact can be used to get an estimate of the spatial resolution of the camera at its current pose.

Spatial resolution is a measure of the smallest size of detail that a sensor can pick up. For a camera, typically Ground Sampling Distance (GSD) is used which measures the distance between the points of ground plane intersection between consecutive pixel centers in the image. The standard equation for GSD is:

$$\text{GSD} = \frac{PD}{f\cos(\theta)} \tag{3.10}$$

where f is the focal length, D is the straight line distance from the camera to the point on the ground where GSD is being measured, P is the distance between pixel centers on the sensor, and θ is the pitch of the camera from nadir [12]. It is important to note that the spatial resolution varies throughout the sensor footprint. In addition, Equation (3.10) does not account for any lens distortion effects.

The following method can be used to estimate both the sensor footprint and the spatial resolution throughout the entire sensor footprint. For each pixel (u, v) in an image, calculate its point of intersection on the ground plane using the raycasting method described above. We then define a rectangular grid containing $1\,\text{m} \times 1\,\text{m}$ cells, whose dimensions are determined by the extrema of the ground plane intersection points, ensuring the grid is large enough to contain the entire sensor footprint. For each grid cell p, we calculate the number of pixels whose ground plane intersection is contained within p; we call this value r for each cell. Algorithm 3.1 gives an overview of how to calculate the number of pixel – ground plane intersections in every cell of the camera's sensor footprint. Note that this map directly gives a measure of px/m^2. To convert this to a measure for the estimate of the spatial resolution throughout the sensor footprint, for each cell we calculate $g = 100\sqrt{1/r}$, where g is the estimate of the spatial resolution in each cell in cm/px.

Algorithm 3.1 Calculation of Pixel Intersection Map

1: Calculate ground plane intersection point for all pixels in image
2: Create grid that covers the entire sensor footprint
3: Bin all ground plane intersection points into corresponding grid cells

This method allows for a straightforward calculation of the effective sensor footprint, which we define as the region on the ground that the camera can see that achieves a spatial resolution threshold. For cameras not oriented facing nadir, there can be a significant difference in the area on the ground achieving different spatial resolution thresholds. For a given spatial resolution G in cm/px, we can calculate the effective sensor footprint by determining which cells in the grid achieve a spatial resolution estimate whose value is less than or equal to the threshold G. Note that a lower spatial resolution value results in a higher granularity of detail in our imagery. Another benefit of this method is that it is able to account for any lens distortion effects since we model the path of each light ray directly.

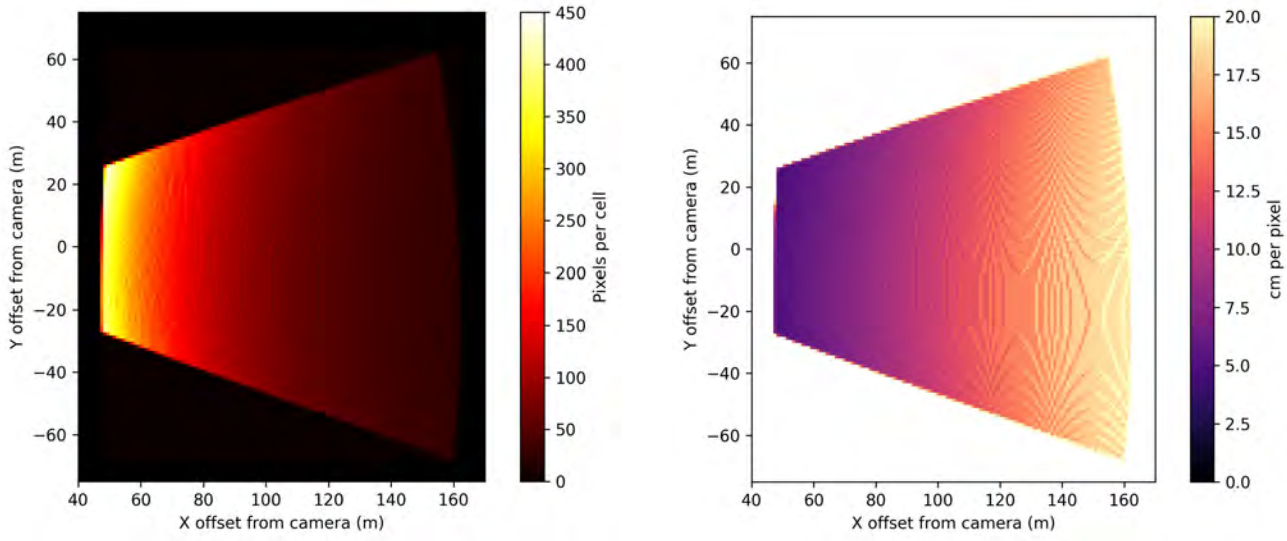
Given a spatial resolution threshold G in cm/px and a grid of ground plane intersection counts, if our goal is just to calculate the effective sensor footprint, we can decrease the number of computations required by flipping the threshold. Instead of converting the number of ground plane intersections r in each grid cell from px/m^2 to cm/px, we can just convert G into a threshold of px/m^2. That is, let $R = (100/G)^2$, where R is the threshold for the minimum number of ground plane intersections each cell must achieve in px/m^2. Algorithm 3.2 below gives an overview of calculating the effective sensor footprint.

Algorithm 3.2 Effective Sensor Footprint Calculation

1: Use Algorithm 3.1 to calculate the pixel intersection map
2: Convert spatial resolution threshold G into ground plane intersection threshold R
3: Select grid cells that achieve a minimum of R ground plane intersections

Figure 3.1 compares the pixel density map to the spatial resolution map from the Grazer-A hi-res

camera, seen in Appendix A, at an altitude of 46 meters and a pitch of $60°$ from nadir to the direction of travel. We can see that the lens distortion effects show up the same in both maps, seen by the curved sampling lines and the lack of symmetry in the sensor footprint. In addition, even though the units measured in each cell differ, we can see a similar pattern for the distribution of data in each plot with a higher density of coverage closer to the camera that dissipates with distance.



(a) Pixel Density

(b) Spatial resolution

Figure 3.1: Sensor footprint of Grazer-A hi-res camera at an altitude of 46 meters and a pitch of $60°$ from nadir to the direction of travel.

3.2 Problem Formulation

3.2.1 Vantage Point Definition

A vantage point is a waypoint that the UAV will be stationary at for a short duration in order to capture imagery. Each vantage point is defined by an (x, y, z) position and an orientation given by a yaw angle. Note that in the path planning stage we assume the roll and pitch of the UAV pose at each vantage point is $0°$, but the pitch of the UAV's onboard camera is considered during path planning.

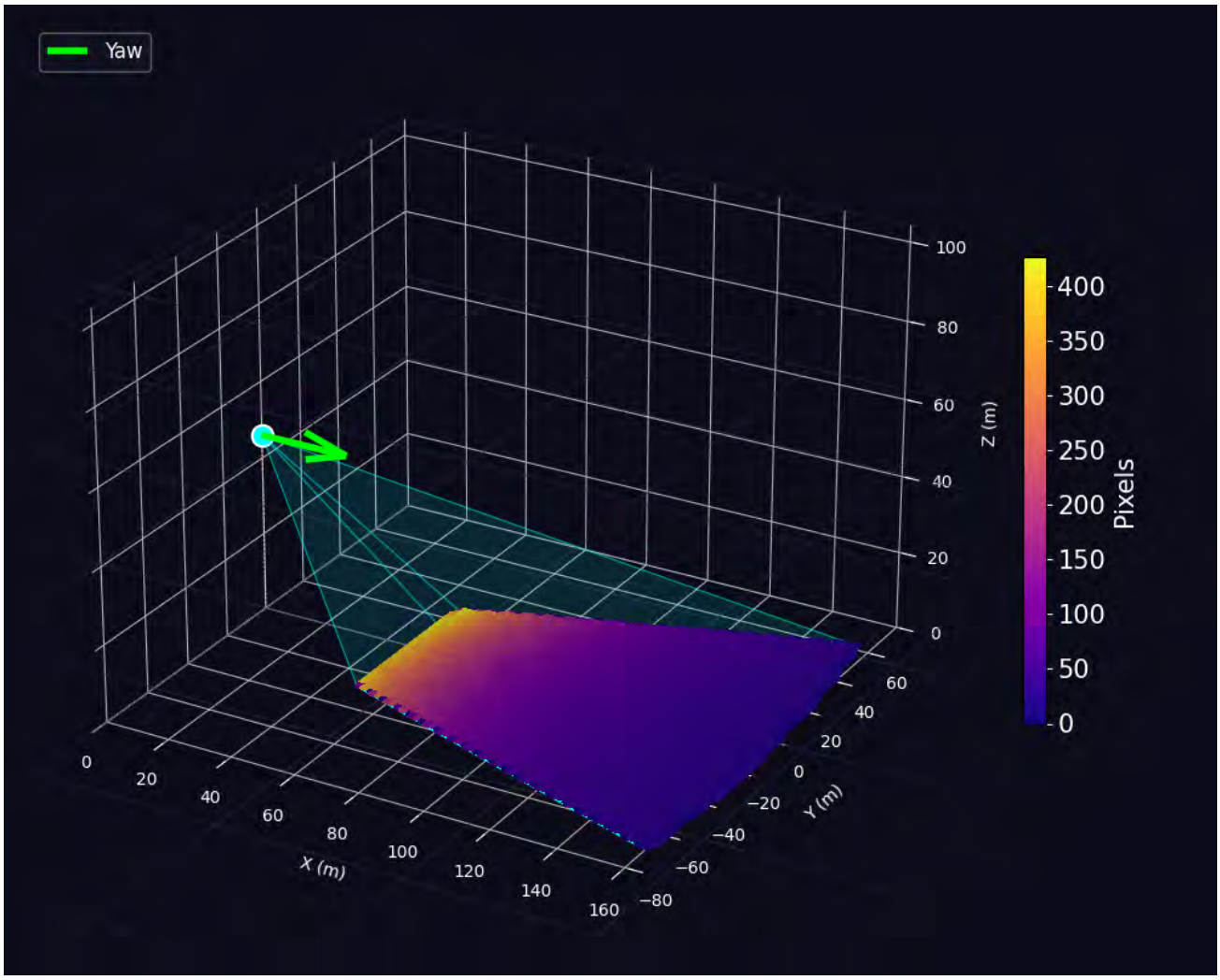


Figure 3.2: Example vantage point and pixel density map using Grazer-A hi-res Camera at an altitude of 46 meters and a pitch of 60° from nadir to the direction of travel.

Figure 3.2 shows an example vantage point defined by the blue dot for its position and the green arrow for yaw and the accompanying ground plane intersection density map.

3.2.2 Problem Statement

Given a 2D search domain defined by a set of perimeter vertices, a calibrated camera attached to a UAV with a known orientation relative to the UAV, a desired search altitude z^*, and a spatial resolution threshold G in cm/pixel, we want to generate a set of vantage points such that every $1\,\text{m} \times$

1 m cell in the search domain achieves a minimum number of pixel intersections with the ground plane from the sensor footprint of at least one vantage point. The number of pixel intersections per cell R is calculated via $R = (100/G)^2$. When calculating ground plane intersections for the sensor footprint, we assume a flat Earth and that the search domain we are trying to cover is occlusion-free, a known oversimplification for many environments. This set of vantage points should maximize the efficiency of coverage by minimizing the total time to cover the search domain, minimizing the amount of coverage that is wasted outside of the search domain, and by minimizing the portion of the search domain that is covered by multiple vantage points as opposed to a single vantage point. These goals for an efficient coverage path plan can be achieved by minimizing the number of vantage points selected to cover the entire search domain.

3.2.3 Integer Linear Program Formulation

We model the task of generating vantage points using the Set Cover Problem, where the universe of elements that we are trying to cover is all of the cells in the cellular decomposition of the search domain, which divides the search domain into a homogeneous grid of $1\,\text{m} \times 1\,\text{m}$ square cells. The family of subsets is the effective sensor footprints from a set of candidate vantage points. These candidate vantage points are a discretization of all of the possible vantage points throughout the search domain. We say that a cell is covered if it receives R pixel intersections from the sensor footprint of at least one vantage point. Note that we do not sum the number of pixel intersections for a given cell between vantage points because the relationship between the number of pixels-on-target and quality of detection is not a linear relationship [9]. For example, a cell receiving 100 pixel intersections equally distributed between 5 vantage points does not result in the same quality of detection as a cell receiving

100 pixel intersections from a single vantage point.

Formally, let P be all of the cells in the cellular decomposition of the search domain and define $C = \{1, \ldots, m\}$ as the set of indices representing our candidate vantage points. Let $SF(j) \subseteq P$ be the cells of the search domain in the sensor footprint of vantage point j that achieve a minimum of R ground plane intersections from the camera's image pixels. For each cell p in our search domain, define N_p as the set of vantage points that can be chosen to cover cell p, i.e.,

$$N_p = \{j \in C \mid p \in SF(j)\} \tag{3.11}$$

Equation (3.12) below shows the formulation of the Set Cover Problem as an ILP to generate the minimum cardinality set of vantage points such that every cell in the search domain satisfies the minimum coverage threshold, i.e.,

$$\text{Minimize} \quad \sum_{j=1}^{m} x_j \tag{3.12a}$$

$$\text{Subject to} \quad \sum_{j \in N_p} x_j \geq 1 \qquad \forall p \in P \tag{3.12b}$$

$$x_j \in \{0, 1\} \qquad \forall j \in C \tag{3.12c}$$

Here, x_j is a binary variable representing if vantage point j is selected to cover the search domain, where Equation (3.12c) restricts x_j to be binary valued. The objective function in Equation (3.12a) minimizes the number of vantage points selected and Equation (3.12b) ensures that every cell in the search domain is covered by the effective sensor footprint of at least one selected vantage point.

3.3 Solving Methods

3.3.1 Setup

The finite set of candidate vantage points C is composed by sampling variations of (x, y) position and yaw within the search domain, holding the altitude z^* constant across the entire set. It is sufficient to sample points every Δ_x meters in the x-direction, every Δ_y meters in the y-direction, and n_ψ evenly distributed yaws at each point, to form a grid of candidate vantage points, where Δ_x, Δ_y, and n_ψ are user-defined parameters. The density of sampling gives a tradeoff between the speed of solving Equation (3.12) and the precision of the optimal vantage point placement.

Given two candidate vantage point indices $i, j \in C$, the maps of the number of pixel ground plane intersections in each cell of their sensor footprints, $SF(i)$ and $SF(j)$, are equivalent under rotation and translation in the XY-plane. For each pixel in the camera, the ray in the camera frame that passes through that pixel and the center of projection is invariant to the vantage point, the only difference in the calculation of the sensor footprints is the extrinsic matrix used to transform the ray from the camera frame to the world frame. Since altitude is held constant across all candidate vantage points, the only quantities that vary between any two extrinsic matrices of vantage points i and j is the (x, y) position and yaw. Note that the yaw is encoded in the rotation matrix ${}^cR^w$. Therefore $SF(i)$ and $SF(j)$ differ only by rotation and translation in the XY-plane, corresponding to the difference in yaw and (x, y) position of the two vantage points.

This motivates the use of a template pixel intersection map, which contains the number of ground plane intersections in each cell of the sensor footprint from an example vantage point, which can then be transformed to the pixel intersection map for any candidate vantage point j, allowing us to

only calculate the ground plane intersections for a single vantage point. Consequently, this approach greatly reduces the number of computations required for calculating the pixel intersection map for every candidate vantage point. For example, consider the Grazer-A hi-res camera, whose calibration is shown in Appendix A. Since this camera has a resolution of 1280×768, we need to calculate $1280 \times 768 = 983,040$ ground plane intersections using Algorithm 3.1 to calculate the pixel intersection map. Repeating this approach for every single candidate vantage point would be very computationally intensive. Instead, after computing the template pixel intersection map, for each candidate vantage point j, we can create a transformation matrix that transforms the template pixel intersection map to the pixel intersection map of vantage point j.

We use the point $(0, 0, z^*)$ with a yaw of 0° as the vantage point used to construct the template pixel intersection map using Algorithm 3.1. We then construct a matrix F_0 to store the template pixel intersection map, where each column is a vector storing the cell center in homogeneous coordinates and also storing the number of pixel intersections in that cell. For example, if a cell with center (x, y) has r ground plane intersections from the camera, we construct the vector shown in Equation (3.13) to be a column in F_0.

$$\begin{bmatrix} x & y & 1 & r \end{bmatrix}^T \tag{3.13}$$

For each candidate vantage point $j \in C$, we construct a transformation matrix T_j, shown in Equation (3.14), that transforms the template pixel intersection map to the pixel intersection map at candidate vantage point j located at (x, y, z^*) with a yaw of ψ radians. T_j is a standard 2D transformation matrix that rotates points around the origin by ψ and performs a translation of (x, y), but is embedded in a 4×4 matrix. Since the fourth row of T_j is $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$, based on the columns

of F_0, for each cell center that is transformed by T_j, the number of pixel intersections is unmodified.

$$T_j = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & x & 0 \\ \sin(\psi) & \cos(\psi) & y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

Thus, in order to calculate $SF(j)$ for all $j \in C$, we can use Algorithm 3.3, which requires computing only a single template pixel intersection map rather than computing $|C|$ pixel intersection maps if we naively used Algorithm 3.1. Note that when using Algorithm 3.3, we take $n_\psi = 4$ corresponding to yaws of 0, $\frac{\pi}{2}$, π, and $\frac{3\pi}{2}$ radians, since axis-aligned rotations ensure that transformed cell centers map exactly onto other grid cells, avoiding the effects of interpolation that can exist when rotating by non-axis-aligned angles.

Algorithm 3.3 Improved Effective Sensor Footprint Calculation

1: Use Algorithm 3.1 to calculate the template pixel intersection map F_0
2: Convert spatial resolution threshold G into ground plane intersection threshold R
3: **for** each candidate vantage point $j \in C$ **do**
4: Compute $F_j = T_j F_0$
5: Compute $SF(j)$ by selecting cells in F_j achieving at least R ground plane intersections
6: **end for**

Even with using Algorithm 3.3, generating the F_j for every candidate vantage point j still can serve as a bottleneck in the generation of vantage points. Since each computation of $F_j = T_j F_0$ is independent, we can use parallelization to compute each F_j asynchronously, greatly improving total runtime.

3.3.2 Vantage Point Selection Methods

CP-SAT can be used to solve Equation (3.12), though finding a high-quality solution typically requires at least one minute of solving time, making it unsuitable for real-time applications but viable when some planning time is available, even if limited. Empirically, for modestly sized problems, it can take over an hour to find an optimal solution but a good feasible solution can be found in a few minutes. When only giving the solver a few seconds to work, the solver frequently is unable to find a feasible solution.

For real-time solving, a greedy algorithm can be used to generate an approximate solution [25]. While there is no guarantee that the cardinality of the selected set of vantage points is minimal, the greedy solver is guaranteed to return a feasible solution if a feasible solution exists. The main motivation for using a greedy algorithm is the relatively quick solving time, typically on the scale of a few seconds, which makes it a good fit for real-time path planning when the goal is to start search operations as quickly as possible, rather than wait for an optimal solution.

At a high level, this algorithm keeps selecting the vantage point that will cover the largest number of cells not already covered by the previously selected vantage points. We begin by initializing a set U to be $U = P$, where P is all of the cells in the search domain. We also create a set V, initialized to $\emptyset$, used to store the selected vantage points. U represents the cells that have not been covered by any vantage point $v \in V$. While U is nonempty, we select the vantage point $j \in C$ whose sensor footprint covers the most cells $p \in U$. Formally, select the vantage point v^* where

$$v^* = \underset{j \in C}{\operatorname{argmax}} \, |SF(j) \cap U|$$

We then set $U \leftarrow U \setminus SF(v^*)$, $C = C \setminus \{v^*\}$, and $V = V \cup \{v^*\}$. I.e., we remove all of the cells in U that vantage point v^* covers, remove v^* from our set of candidate vantage points that we can select from, and add v^* to our set of selected vantage points. Algorithm 3.4 below summarizes the greedy vantage point selection.

Algorithm 3.4 Greedy Vantage Point Selection

$U \leftarrow P$
$V \leftarrow \emptyset$
while $U \neq \emptyset$ **do**
    $v^* \leftarrow \text{argmax}_{j \in C} \ |SF(j) \cap U|$
    $U \leftarrow U \setminus SF(v^*)$
    $C \leftarrow C \setminus \{v^*\}$
    $V \leftarrow V \cup \{v^*\}$
end while
return V

Algorithm 3.5 below gives an overview of generating and selecting a set of vantage points such that every cell in the search domain satisfies the coverage threshold. One important thing to note is that these vantage points are unordered, motivating the work in Chapter 4.

Algorithm 3.5 Vantage Point Generation

Perform cellular decomposition of search domain
Generate set of candidate vantage points C using grid-based sampling
Calculate $SF(j)$ for all $j \in C$ using Algorithm 3.3
Solve Set Cover Problem formulated in Equation (3.12)

Chapter 4: Vantage Point Sequencing

This chapter outlines methods for determining the sequence of vantage points the UAV to travel, with particular emphasis on temporal considerations. Section 4.1 describes a VRP that can be used to maximize coverage within a time limit. Section 4.2 describes methods for generating a route in real-time and Section 4.3 discusses the considerations for extending these algorithms to dynamic replanning.

4.1 Coverage Orienteering Problem

4.1.1 Motivation

In a simple case without any constraints on mission time, using the TSP to sequence all selected vantage points will result in the most efficient coverage path plan possible. The solution of the TSP will minimize the total travel time over all possible permutations of sequences of the selected vantage points. However, if we do have a maximum mission time constraint, it may not be physically possible to travel and perform inspection at all of the selected vantage points. For instance, if a UAV has a limited battery, it may not be possible to complete the entire search operation before the battery depletes. In this case, an operator may prefer to cover only a portion of the search domain and maintain their desired spatial resolution threshold rather than utilize a sparser spatial resolution threshold. Reducing the coverage threshold would likely decrease the number of vantage points and thus decrease the total time to cover the search domain, but this could also harm the probability of detection due to achieving fewer pixels-on-target in most cells. Thus, we want a way to select and sequence vantage points that can maximize the amount of coverage within a maximum mission time.

This objective initially leads to the Orienteering Problem (OP) since the goal of the OP is to maximize the reward obtained within a time limit. For our application, we could use the amount of coverage from each vantage point as the reward for that vantage point. However, as discussed in Section 2.2, the standard OP can only handle rewards that are independent of the rewards of any other vantage point. For our problem, independent rewards fail to accurately quantify the total coverage throughout the search domain due to overlapping coverage. Even though the generation of vantage points from Algorithm 3.5 implicitly reduces overlapping coverage, in practice some overlapping coverage is inevitable due to the discretization of candidate vantage points and the non-uniformity of the UAV's sensor footprint. When selecting a subset of vantage points, the amount of additional coverage any given vantage point will add depends on the coverage of all of the other selected vantage points.

One potential remedy to this could be using a reward structure similar to the Target Visitation Problem, where the reward for each vantage point is based on the pairwise relative ordering of vantage points. However, this structure does not correctly handle unique coverage since unique coverage is not a pairwise relationship dependent on which vantage point we visit first but is instead dependent on the entire set of vantage points selected. Thus, the reward should be tied to the cells in the search domain rather than the vantage points themselves in order to correctly account for the interdependence of coverage from different vantage points.

This motivates the Coverage Orienteering Problem, where the objective is to maximize the number of cells in the search domain that are covered by at least one vantage point. After we independently developed the formulation in Section 4.1.2, basing flow and MTZ constraints on the formulation in [41], we discovered the CCOP, discussed in Section 2.2, which uses a very similar reward structure for unique coverage. The primary distinction is that the CCOP's domain-specific

constraints for water sampling with USVs, in a carrying capacity constraint and mandatory cluster visit constraint, have no analog in our setting.

4.1.2 Formulation

Given a set of vantage points C', a maximum mission time $T_{\max}$, a vantage point inspection duration s held constant across all vantage points, and an estimate for the travel time $t_{i,j}$ between each pair of vantage points $i, j \in C'$, we formulate the Coverage Orienteering Problem in Equation (4.1) with the goal of maximizing the number of cells in the search domain that are covered by at least one selected vantage point within $T_{\max}$. For each cell p in the search domain P, let y_p be a binary variable indicating whether cell p is sufficiently covered by at least one vantage point selected in the routing. Denote $d \in C'$ as the depot vantage point which we will start our search from and must return to before $T_{\max}$ and let $C^* = C' \setminus \{d\}$. Note that we perform inspection at d at the start of the mission but not on our return.

While Equation (4.1) is generalized to work for any set of vantage points, C' is intended to be the set of vantage points generated by Algorithm 3.5 where $C' \subseteq C$. The variables in this formulation are chosen to be consistent with the notation established in Equation (3.12). Most notably, we use $v_{i,j}$ instead of $x_{i,j}$ to represent the direct path between vantage points i and j since x_j represents if a vantage point is selected to cover the search domain in Equation (3.12).

Equation (4.1) below shows the formulation of the Coverage Orienteering Problem. In this formulation, the objective function in Equation (4.1a) maximizes the number of cells that are sufficiently covered. Equation (4.1b) constrains $v_{i,j}$ to be a binary variable, which represents if the UAV travels directly from vantage point i to vantage point j. Similarly, Equation (4.1c) forces z_j to be a binary

variable representing if vantage point j is visited. Combined with the constraints from Equation (4.1h), if a vantage point is selected then we only travel to it once and only travel from it once, aside from the depot. Equation (4.1g) handles these constraints for the depot.

$$\text{Maximize} \quad \sum_{p \in P} y_p \tag{4.1a}$$

$$\text{Subject to} \quad v_{i,j} \in \{0,1\} \qquad \forall i \neq j \in C' \tag{4.1b}$$

$$z_j \in \{0,1\} \qquad \forall j \in C' \tag{4.1c}$$

$$y_p \in \{0,1\} \qquad \forall p \in P \tag{4.1d}$$

$$y_p \leq \sum_{j \in C' \cap N_p} z_j \qquad \forall p \in P \tag{4.1e}$$

$$y_p \geq z_j \qquad \forall j \in C' \cap N_p, \forall p \in P \tag{4.1f}$$

$$z_d = \sum_{j \in C^*} v_{d,j} = \sum_{i \in C^*} v_{i,d} = 1 \tag{4.1g}$$

$$z_k = \sum_{i \in C'} v_{i,k} = \sum_{j \in C'} v_{k,j} \qquad \forall k \in C^* \tag{4.1h}$$

$$\sum_{i \in C'} \sum_{j \in C'} v_{i,j}(t_{i,j} + s) \leq T_{\max} \tag{4.1i}$$

$$u_i - u_j + 1 \leq (|C'| - 1)(1 - v_{i,j}) \qquad \forall i, j \in C^* \tag{4.1j}$$

$$2 \leq u_j \leq |C'| - 1 \qquad \forall j \in C^* \tag{4.1k}$$

For a given cell p, we use the variable y_p to denote if it is covered by a selected vantage point using the constraints in Equations (4.1e), (4.1f), and (4.1d). If there are no vantage points selected that cover cell p, then $z_j = 0$ for all $j \in C' \cap N_p$, making the right-hand side of Equation (4.1e) equal to

0, giving us $y_p \leq 0$. Based on Equation (4.1d), we get $y_p = 0$. On the other hand, suppose we have at least one selected vantage point that covers p. By Equation (4.1f) we then get at least one constraint $y_p \geq 1$ and we know the minimum value possible for the summation in Equation (4.1e) is then 1. Since y_p is binary, we then get $y_p = 1$. Thus, this group of constraints allows us to measure if a cell is covered by at least one vantage point while also not double-counting overlapping coverage.

Equation (4.1i) is the constraint that ensures the mission is completed within a duration $T_{\max}$. For a potential route, this constraint sums up all of the travel time between edges along with the total amount of time performing inspection at each vantage point. Meanwhile, Equations (4.1j) and (4.1k) are used to eliminate subtours using the MTZ constraints and variables.

When comparing this formulation to the CCOP, the CCOP does not have an equivalent lower bound constraint to Equation (4.1f). While this does not make a difference for the optimal solution, as the solver will always try to maximize y_p due to its role with the objective function, for our use case when we often only care about the best feasible solution after a few minutes of solving, the explicit lower bound constraint improves the accuracy of the objective function in scoring feasible solutions. Thus, this lower bound constraint helps to improve intermediate best feasible solutions, resulting in coverage path plans that can cover more of the search domain.

4.2 Real-Time Routing Algorithms

Directly solving Equation (4.1) using CP-SAT is not a practical method for real-time operations since it often takes several seconds for the solver to even find a feasible solution for moderately sized search domains of a few acres. In this section, we present methods to perform routing in real-time by solving for approximate solutions to the Coverage Orienteering Problem.

4.2.1 Real-Time Routing

Rather than wait for CP-SAT to find a feasible solution, what if we can buy ourselves computational time? We propose a method that exploits the UAV's flight time to the first vantage point as additional computational time that would not exist if the complete coverage path plan was needed before the mission started. This method uses a heuristic to select the first vantage point the UAV will visit, fixing it as the depot, then using the time from the mission starting until inspection is finished at the depot to solve a VRP to generate a complete coverage path plan. Note that an actual depot still exists but for the purposes of ILP, the depot is now the first vantage point and inspection occurs before the coverage path plan obtained by solving the ILP starts. While this framework is designed to solve Equation (4.1), this framework is applicable to work for other VRPs too.

This framework begins by using the greedy set cover solver from Algorithm 3.4 to generate vantage points, primarily used due to its quickness obtaining a feasible solution. Once the set of vantage points is generated, we use a simple heuristic to decide what is the best vantage point to travel to first. Based on the location of the UAV at the time of vantage point generation, we denote the closest vantage point as the starting depot in our ILP formulation, motivated by the idea that selecting this vantage point minimizes the initial travel time while also immediately contributing to search domain coverage. In practice, this approach typically provides an additional ten to fifteen seconds of computational time available for completing the coverage path plan.

After selecting the depot, we start an asynchronous thread that will be used to solve the ILP. The solver is given a maximum solving time T_s, where $T_s = 0.95 \cdot (t_0 + s)$ and t_0 is the time to travel from the UAV's current location to the first vantage point. After starting the solver, the UAV will start flying to the first vantage point to perform inspection. Once inspection at the depot is complete, the best

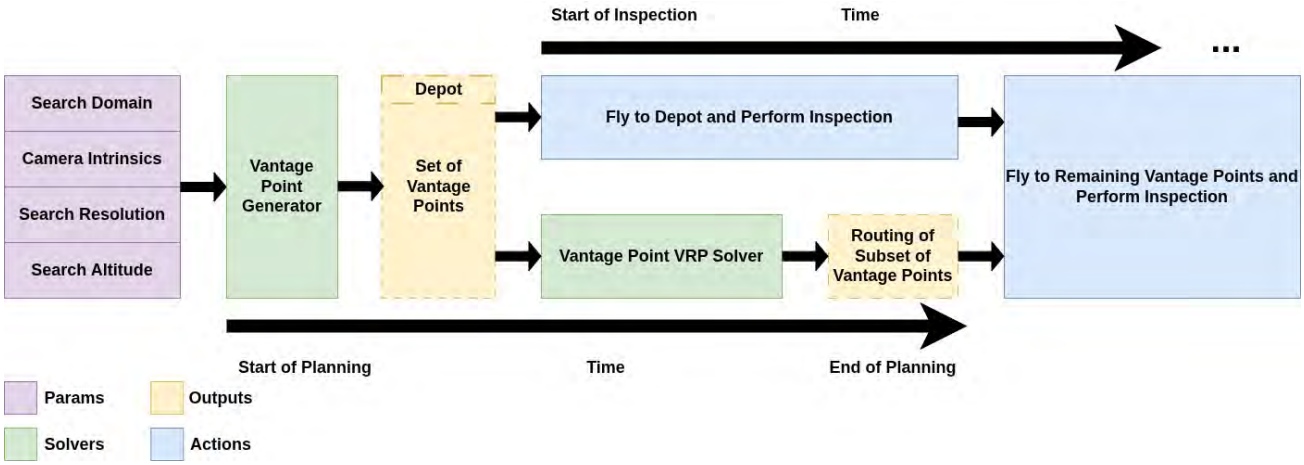


Figure 4.1: Framework for solving VRPs for real-time routing of vantage point-based coverage path plans.

feasible solution is retrieved from the ILP solver and the UAV executes the remainder of the coverage path plan. Figure 4.1 gives an overview of this framework, showing what parts execute in parallel and how they sync up after inspection at the depot is complete.

4.2.2 Solving Methods

Although the Coverage Orienteering Problem is a VRP, its reward structure cannot be implemented using the OR-Tools Vehicle Routing Library. For an exact solver, we instead use CP-SAT. While the travel time estimates are real-valued, CP-SAT requires all variables to be integer-valued. Since the travel times are already an approximation, which are the only non-integer valued variables in the formulation of Equation (4.1), rounding travel times to integers does not have a notable effect on the modeling of the problem.

Large Neighborhood Search (LNS), originally proposed for solving VRPs, is a metaheuristic that combines characteristics of local search with techniques for solving constraint optimization problems, balancing adherence to constraints while focusing the search space to higher quality solutions [57–59].

Until some loop termination condition is met, given the best current solution as a list of variables, we use a neighborhood heuristic to determine which variables we want to explore, then unassign those variables in the current solution, referred to as the destroy step. We then search for what new assignment of those variables is an improvement over the best solution and also satisfies all of the constraints, referred to as the repair step.

For the implementation of the LNS solver, we begin by constructing an initial feasible solution by greedily selecting the vantage point that covers the largest number of previously uncovered cells, then finding the point of insertion in the routing that minimizes the increase in travel time from adding a new vantage point. We continue adding vantage points until every possible addition of any of the remaining vantage points would violate $T_{\max}$. Note that this greedy selection is separate from Algorithm 3.4 due to the temporal consideration. We use a random destroy operator and a greedy repair operator. The destroy operator randomly removes k vantage points from the current solution, while the repair operator greedily reinserts vantage points based on the marginal coverage gain with respect to $T_{\max}$. k is a user-defined parameter for the size of the neighborhood. We take $k = 3$. At the end of each iteration, we determine the best solution by the number of cells covered. Algorithm 4.1 below gives an overview of using LNS to solve the Coverage Orienteering Problem with a random destroy operator and a greedy repair operator.

4.3 Extension to Dynamic Search

Section 4.2 introduced a framework to extend the computational time available for routing to include the time flying to the first vantage point and inspection at that vantage point, but this can still be a short duration yielding a suboptimal coverage path plan. A better approach is to extend

Algorithm 4.1 LNS for Solving the Coverage Orienteering Problem

1: Greedily construct initial feasible solution
2: **while** total solving time is less than T_s **do**
3: Let the current solution be a copy of the best solution
4: Random Destroy - randomly remove k vantage points from current solution and place them into a list of repair candidates
5: Add k random vantage points not in best solution to repair candidates
6: Greedy Repair - greedily select vantage points from repair candidates to add to current solution subject to $T_{\max}$
7: Update best solution if current solution covers a larger amount of the search domain
8: **end while**
9: **return** best solution found

this framework to support dynamic replanning, where we account for the actual coverage that we have obtained so far in our mission, balancing a quick online planner with an ideally close to optimal coverage path plan. The initial solver considered for dynamic replanning was CP-SAT, where after flying to each vantage point, we update the coverage map and remaining mission time, which ideally would not deviate much from what was planned, allowing us to take advantage of the existing search tree. Unfortunately, CP-SAT does not allow for warm starts, causing the solver to need to build a new search tree whenever the coverage map or remaining mission time is updated. This motivated using LNS since we can continually retrieve the best feasible solution, and it is easy to modify constraints or mission parameters for LNS without restarting. While the full implementation and experimental validation of dynamic replanning is contained in future work, the choice of using LNS in Section 4.2 was made with dynamic search in mind.

Chapter 5: Simulation and Experimental Results

5.1 Experimental Testbed

5.1.1 MAVericks

MAVericks is a Robot Operating System 2 (ROS2) based aerial autonomy stack developed by DEVCOM Army Research Laboratory (ARL) that is meant for small UAS in simulation and in the real world [1, 60]. In particular, MAVericks can run on the ModalAI VOXL2. MAVericks runs both onboard a UAV and on a ground control software (GCS) computer. The key capability from MAVericks for our application of search and rescue is its ability to support search missions beyond visual line of sight and even when communication between a UAV and the ground station is lost.

5.1.2 Hardware

For experimental results, we utilize a quadcopter UAV called the ARL Grazer-A, which is equipped with the ModalAI VOXL2 as its onboard computer and uses a mRo Control Zero H7 flight controller [1, 60]. A picture of the Grazer-A can be seen in Figure 5.1.

For communication between the UAV and ground station, we use a 2.4 GHz radio from Doodle Labs for the data link and a 900 MHz Jeti remote control (RC) transmitter. The Grazer-A is equipped with an onboard high-resolution (hi-res) camera that can be oriented with a pitch between 15° and 60° from nadir to the UAV's direction of travel. It is important to note that this is not a gimbal and is fixed during flight. Calibration info for this camera can be seen in Appendix A. While we examine the effects of different orientations of the camera in this chapter on the sensor footprint, all flight testing results configure the hi-res camera at a pitch of 60° from the ground plane toward the direction of travel. Based on operational experience, we observed an effective battery life of approximately 12–15



Figure 5.1: ARL Grazer-A [1]

minutes per battery.

5.1.3 Ground Station

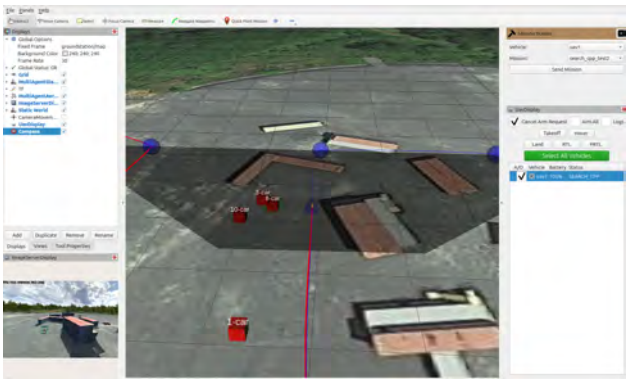
For the ground station, we use a Dell Precision 7680 laptop running Ubuntu 24.04, and also utilize two different GCSs in tandem. QGroundControl (QGC) is an open source GCS meant for MAVLink based platforms [8], which we primarily use to ensure safe operating procedures. QGC is used to set UAV safety parameters like failsafes and geofences in addition to providing a reliable display of the UAV status including telemetry data and battery level. At the same time, we also use the MAVericks GCS to interface with MAVericks running onboard the UAV and send mission plans to execute autonomously on the Grazer-A. The MAVericks GCS uses RViz for its user interface. Figure 5.2 shows a screenshot of QGC and a screenshot of the MAVericks GCS.

5.1.4 Testing Location

Our testing location is the University of Maryland (UMD) UAS Research and Operations Center's (UROC) testing facility located in southern Maryland called Raley Farm, shown in Figure 5.3. This



(a) QGroundControl GCS



(b) MAVericks GCS [9]

Figure 5.2: GCS used for flight testing.



(a) Ground imagery



(b) Satellite imagery [61]

Figure 5.3: Raley Farm test site in southern Maryland.

facility is a 70-acre open field, closely reflecting our occlusion-free environment and flat Earth assumptions from Chapter 3. All flight testing was performed in collaboration with UROC, whose expertise in UAS testing and operational protocols helped to ensure safe flight testing operations.

5.1.5 Simulation

The majority of experimentation was performed using a high visual fidelity Unity-based simulation environment. This simulation environment is part of the MAVericks aerial autonomy stack [60] and is

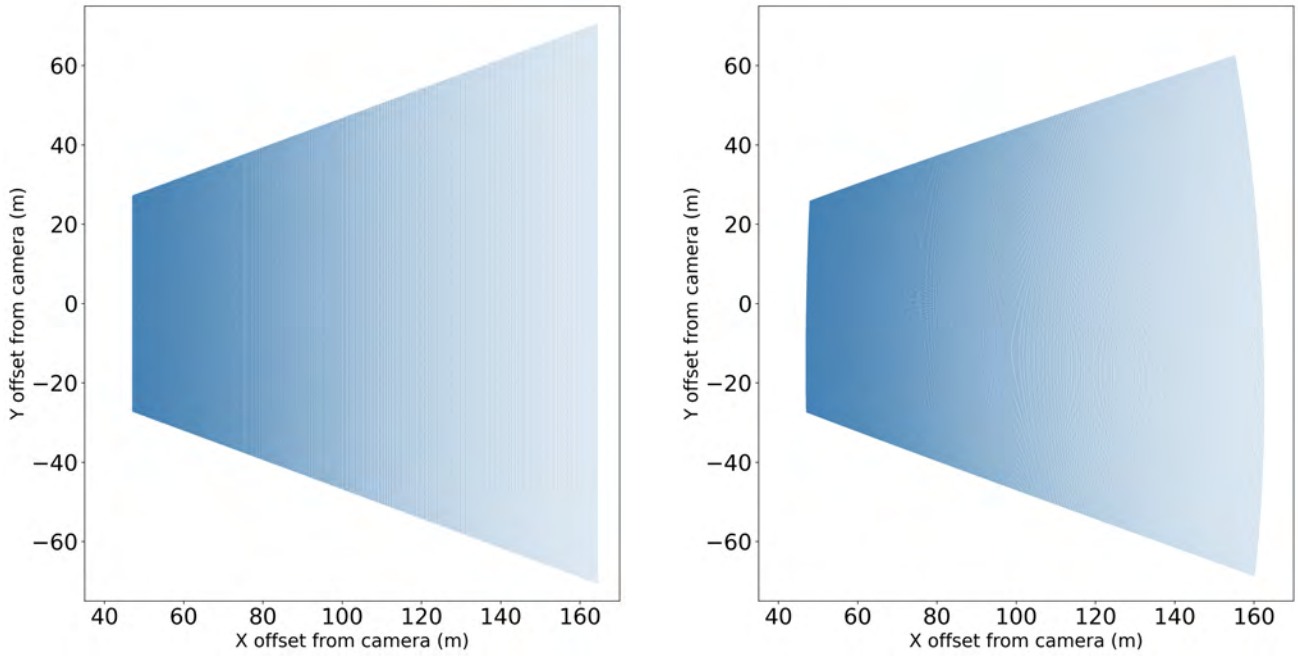


Figure 5.4: MAVericks Unity-based simulation environment depicting a Grazer-A UAV flying at the Raley Farm test facility [1].

suitable for software-in-the-loop testing. Digital twins of the Grazer-A running MAVericks and of the Raley Farm testing facility allow simulation results to match real-world testing as close as possible. Figure 5.4 shows a screenshot of the MAVericks simulation environment.

5.2 Sensor Footprint Analysis

In this section, we analyze how different camera parameters affect the sensor footprint and spatial resolution throughout the sensor footprint of a camera attached onboard a UAV. Calibration parameters for Grazer-A hi-res camera used can be found in Appendix A. Unless otherwise specified, the altitude of the camera is 46 m and the camera is oriented at an angle of $60°$ from nadir to the direction of travel.



(a) No lens distortion

(b) Plumb Bob distortion

Figure 5.5: Projection of all pixels from the Grazer-A hi-res camera to the ground plane at an angle of 60° from nadir to the direction of travel at an altitude of 46 meters.

5.2.1 Distortion Effects

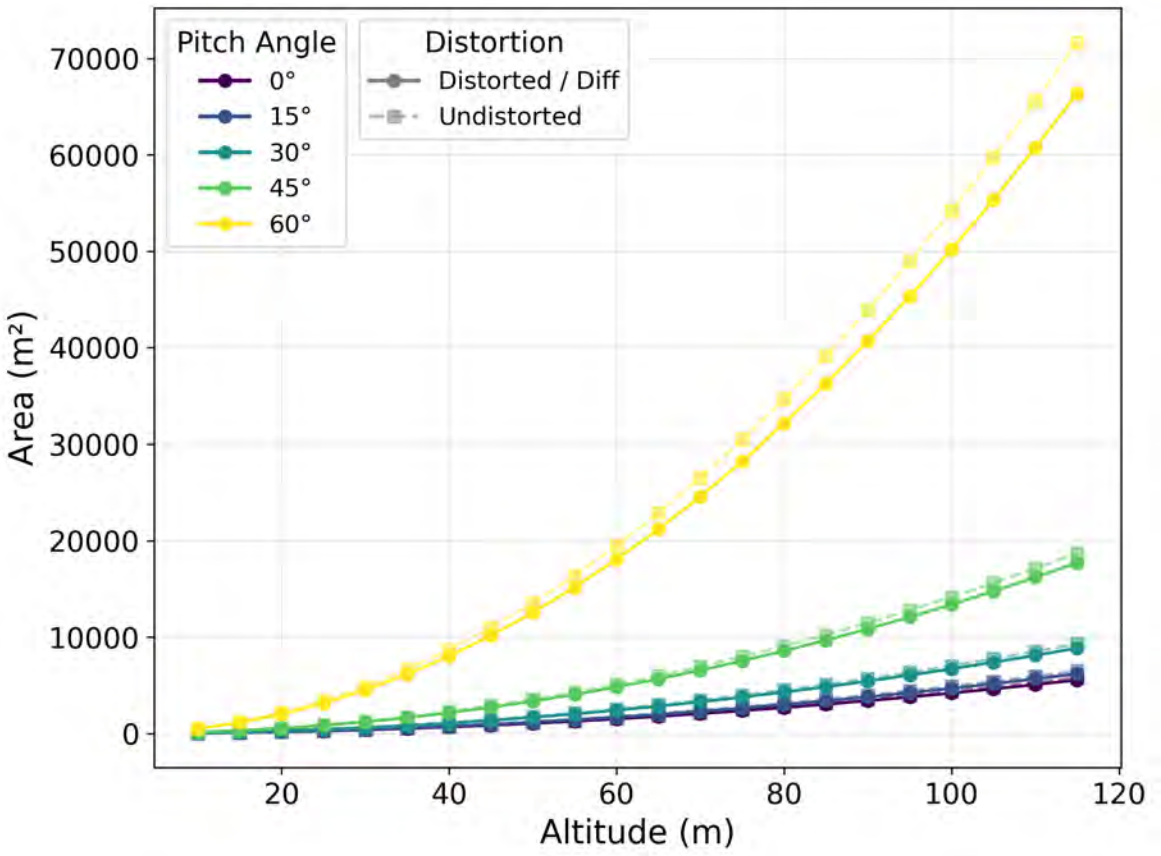
We first begin by exploring the effects of lens distortion on the sensor footprint. Figure 5.5 compares the ground plane intersections for all pixels in the Grazer-A hi-res camera with and without accounting for lens distortion, and before binning intersection points. The rasterization lines visible in the no-distortion plot appear curved in the distortion plot due to radial lens distortion affecting the ground plane intersection points of individual pixels. These lines are image compression artifacts and are not physically meaningful, however, they do a good job of illustrating the effect of lens distortion.

Looking at the overall shape of the sensor footprint, we can see that radial lens distortion deforms the trapezoidal shape of the sensor footprint into a curved shape where the edges of the sensor footprint bow outward. This curvature is the most notable on the far edge of the sensor footprint because based on the Plumb Bob distortion model shown in Equation (3.5), radial distortion is applied to points in the

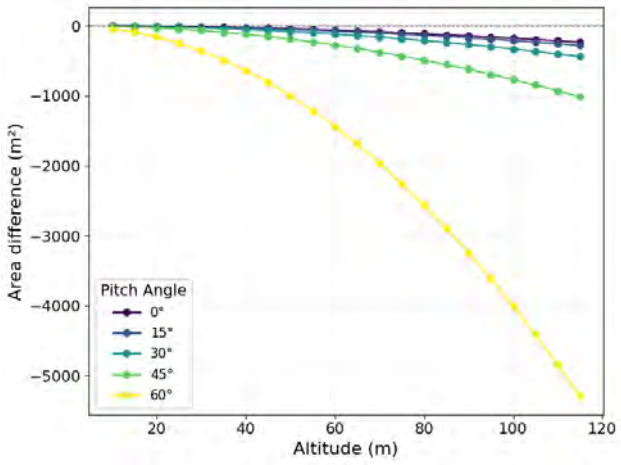
normalized image frame by scalar multiplication by a polynomial factor that increases as the distance from the principal point increases. While not as noticeable as the radial distortion effects, the tangential distortion causes a small rotational skew.

The lens distortion also affects the overall size of the sensor footprint. To calculate the area of the sensor footprint, we calculate the area of the convex hull of all of the ground plane intersections points. Given a set of points, its convex hull is the smallest convex polygon that encapsulates all of the points. Without lens distortion, from our baseline configuration, the area is 11,463.5 m^2, while with lens distortion the area decreases to 10,616.52 m^2. Figure 5.6 shows how this difference in area applies to variations in altitude and camera pitch. As altitude increases, the difference in the area of the sensor footprint also increases quadratically. However, we can see from Figure 5.6c that the relative difference in the area of the sensor footprint is independent of the camera's altitude. On the other hand, as camera pitch increases, the magnitude of the relative difference in the area of the sensor footprint increases, demonstrating that camera pitch magnifies the effect of lens distortion.

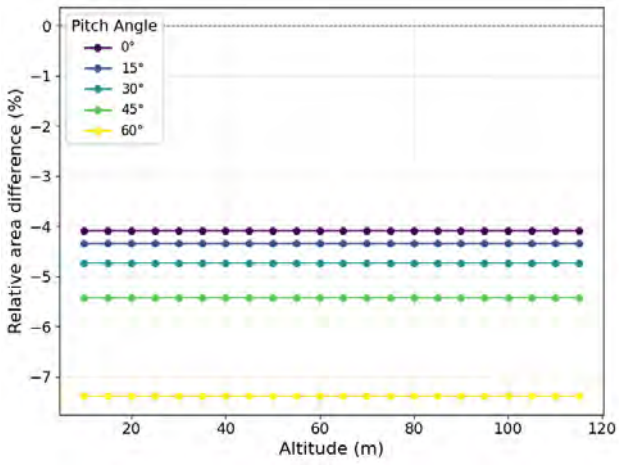
To understand why this occurs, we look at Equation (3.9), which is used as part of the algorithm to calculate the ground plane intersection point for a pixel in the image. A change in the altitude of the camera is reflected only in $\tilde{C}_z$, which is independent of lens distortion. Therefore, altitude just acts as a scalar. However, the denominator of this equation, v_z, is affected by lens distortion as it is calculated from rotating a ray corresponding to a distorted image point by a rotation matrix which includes the camera pitch.



(a) Area of sensor footprint

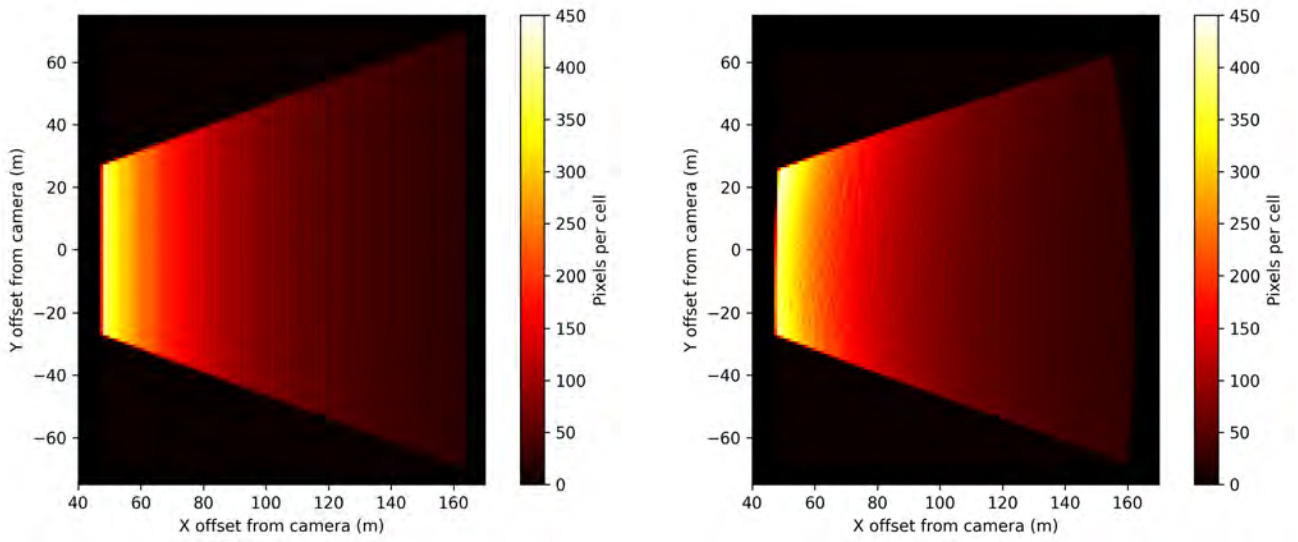


(b) Difference with and without lens distortion



(c) Relative difference in area

Figure 5.6: Effect of lens distortion on the area of Grazer-A hi-res camera's sensor footprint at different altitudes and camera orientations.



(a) No lens distortion

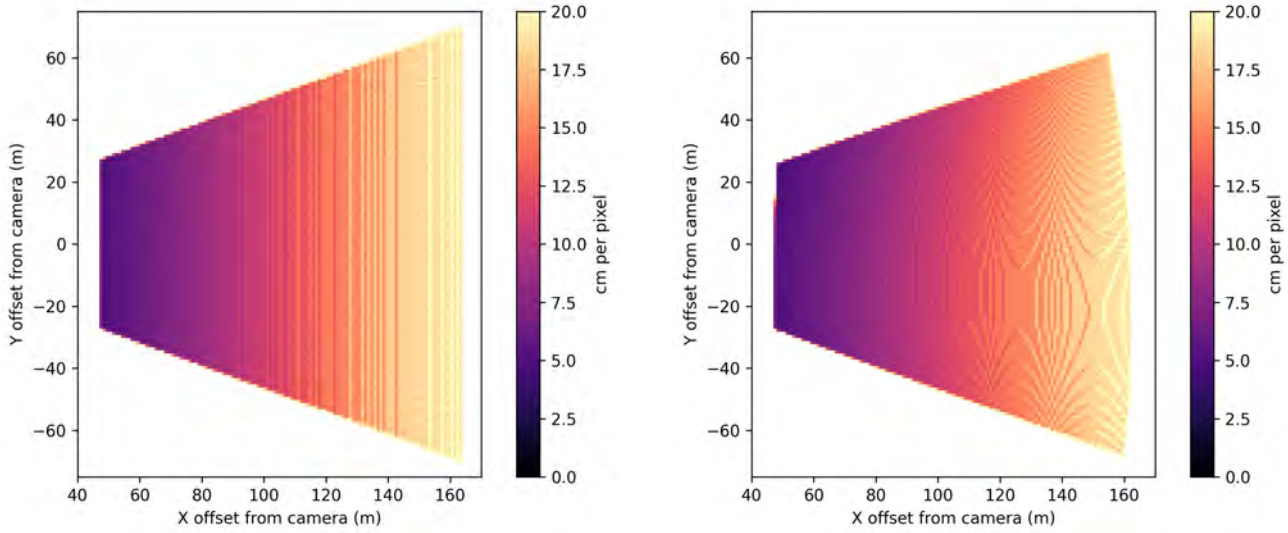
(b) Plumb Bob distortion

Figure 5.7: Pixel density within sensor footprint of the Grazer-A hi-res camera.

5.2.2 Pixel Density

Figures 5.7 compares the density of ground plane intersections with and without lens distortion using a cell size of 1 m. Note that unlike the raw pixel intersection plots, the bands that appear in these plots are due to aliasing from binning ground plane intersection points. Starting with Figure 5.7, which shows the raw pixel density for each camera setup, we can see that the highest density region in either of the two plots, the largest region colored white, occurs inside of the sensor footprint closer to the location of the camera in the plot with distortion. Since the sensor footprint with distortion is smaller, the density of pixels is higher throughout the sensor footprint since the number of pixels in the image did not change. We can also see that the bands of pixel density are also curved by the radial distortion.

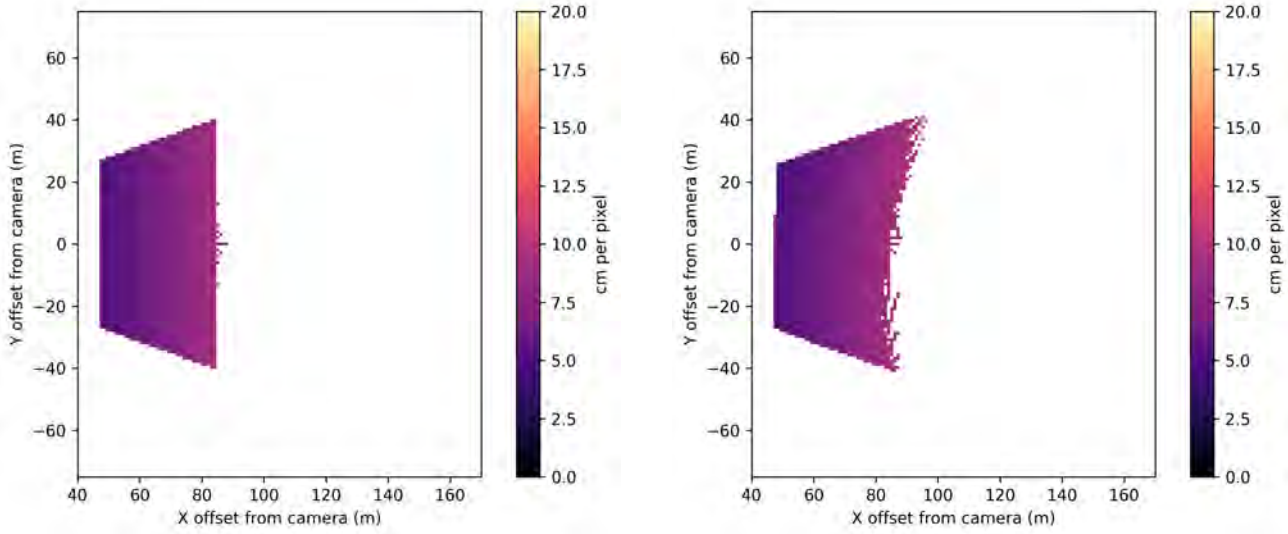
From Figure 5.8, we can see a similar trend in how lens distortion affects the spatial resolution compared to its effect on pixel density. This is expected given the proportional relationship between these two metrics. More importantly for our purposes though, when examining the effective sensor



(a) No lens distortion

(b) Plumb Bob distortion

Figure 5.8: Spatial resolution within sensor footprint.



(a) No lens distortion

(b) Plumb Bob distortion

Figure 5.9: Spatial resolution within effective sensor footprint with spatial resolution threshold of 9.5 cm/px.

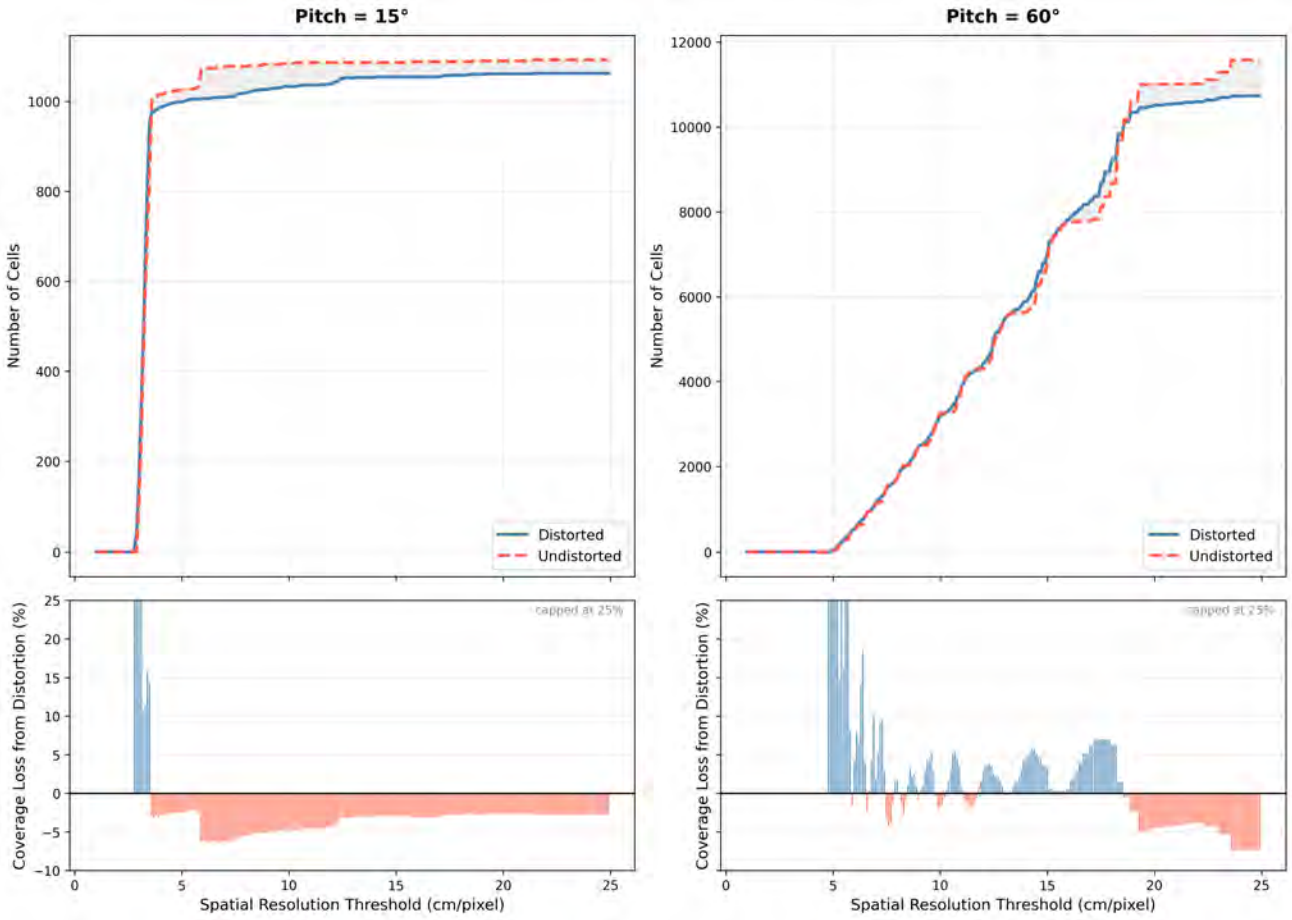


Figure 5.10: Difference in area of effective sensor footprint with and without lens distortion.

footprint from Figure 5.9, the number of cells that meet the spatial resolution threshold of 9.5 cm/px, and the regions those cells exist, is affected by lens distortion. With lens distortion, 2,741 cells meet the spatial resolution threshold while only 2,600 cells meet the threshold with no lens distortion. Figure 5.10 shows how this trend applies to different spatial resolution thresholds and different camera orientations. Note that the altitude was set to 46 m in both plots. At denser spatial resolution thresholds, the distorted sensor footprint typically covers more area, but at sparser thresholds the undistorted sensor footprint covers more area.

Figure 5.11 analyzes the effects of altitude on the size of the effective sensor footprint for different camera pitches. The most notable takeaway from this data is that there is an optimal altitude

for a given spatial resolution threshold, which is what motivates using 46 m as a baseline altitude since that is the optimal altitude to achieve a spatial resolution of 10 cm/px based on maximizing the size of the effective sensor footprint. In addition, for a given spatial resolution threshold, as the camera pitch decreases, the maximum possible size of the effective sensor footprint increases, however, the altitude required to reach this optimal value also increases. This is due to how the image pixels stretch out on the ground plane as the camera pitch increases, causing a larger area to be covered but also a lower density of pixels.

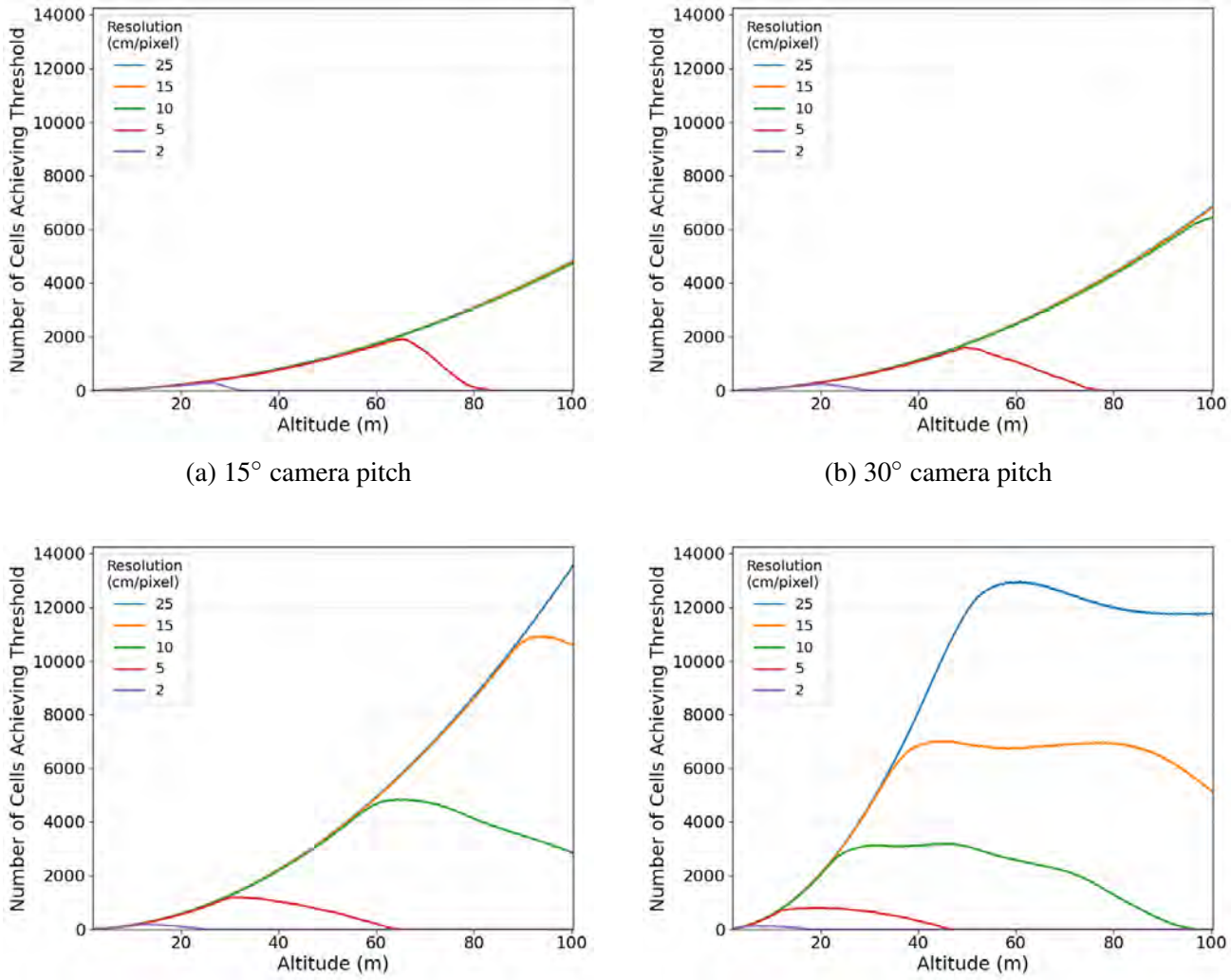
The analysis in this section demonstrates that lens distortion must be accounted for when calculating the sensor footprint and the spatial resolution throughout the sensor footprint. Thus, a more rigorous approach than the GSD calculation from Equation (3.10) is required, proving the necessity of Algorithm 3.1.

5.3 Simulation Results

This section evaluates the efficiency of vantage point coverage path plans using Algorithm 3.5 compared to lawnmower-derived baselines.

5.3.1 Methodology

We compare three different types of coverage path plan: (1) a vantage point plan using the greedy solver from Algorithm 3.4; (2) a vantage point plan using the best feasible solution from CP-SAT after two minutes of solving time; and (3) a lawnmower-derived baseline. The two-minute solving limit was chosen empirically to balance real-world operational usage with near-optimal solutions. Under our testing parameters, we consistently observed an optimality gap of less than 3 within this time limit, which did not improve significantly given close to 30 minutes of solving time. In this context,



(a) 15° camera pitch

(b) 30° camera pitch

(c) 45° camera pitch

(d) 60° camera pitch

Figure 5.11: Comparison of size of effective sensor footprint at different altitudes and camera pitches.

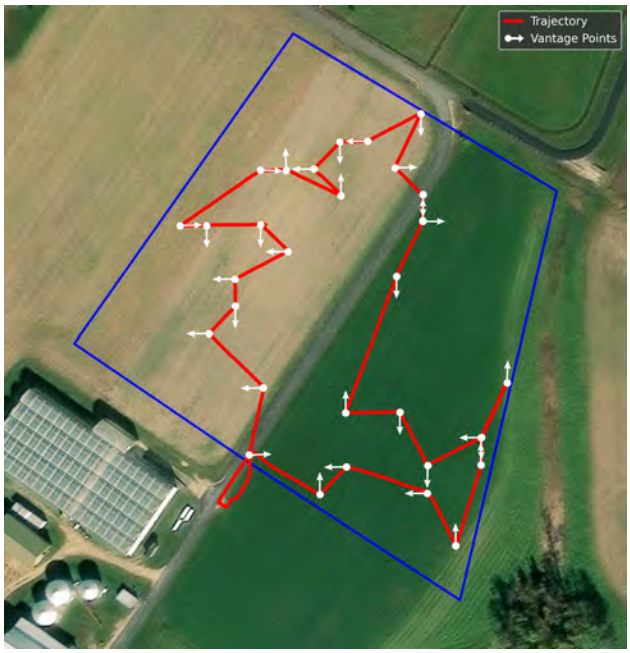
the optimality gap represents the maximum difference in the total number of vantage points between our best feasible solution and the estimated theoretical lower bound which is less than or equal to the value of the optimal solution. Note that ILP solvers like CP-SAT might find an optimal solution but continue solving because the solution has not yet been proven to be optimal, i.e., the optimality gap is greater than or equal to 1.

The search domain, which has an area of 21,906 m^2, was held constant for all flights, including between real-life testing and simulation. Figure 5.12 shows a picture of the search domain and an example of two of the flights flown in real-life testing. We use the mission planner in QGC to draw a search domain defined by a set of waypoints, then save the mission plan as a .plan file. This file is then used alongside the camera calibration parameters and user-defined search parameters to generate the coverage path plan, which is then exported to a .yaml file to be flown in MAVericks. We sampled a total of 812 candidate vantage points, representing four distinct yaws and a step size of 10 m in both the X and Y axes. The search altitude is chosen based on what altitude will have the largest effective sensor footprint. All path planning in this section was performed offline. For the two types of vantage point missions, we use the TSP solver from the OR-Tools Routing Library to perform routing.

For all flights, the camera pitch was 60° from nadir to the direction of travel, transit speed was set to 5 m/s, and the inspection duration at each vantage point was 5 s. This inspection duration was set based on the fact that the controller used often overshot its waypoints or stopped very abruptly so it would take a few seconds to settle at each vantage point. As our baseline coverage path plan, we created a lawnmower-based path plan with built in stops to mimic vantage points, where the altitude was the same as the corresponding vantage point mission at the same spatial resolution threshold. To minimize the total distance, the lawnmower tracks were aligned parallel with the longest edge of the search domain and track spacing was based on width of effective sensor footprint, with the distance



(a) Greedy



(b) CP-SAT best feasible (2 min)

Figure 5.12: Vantage point coverage path plans flown in real-life flight testing at Raley Farm. Flights were conducted at an altitude of 15 m with a spatial resolution threshold of 10 cm/px. Flight plans differ in the method used for solving Equation (3.12).

between stops based on its height.

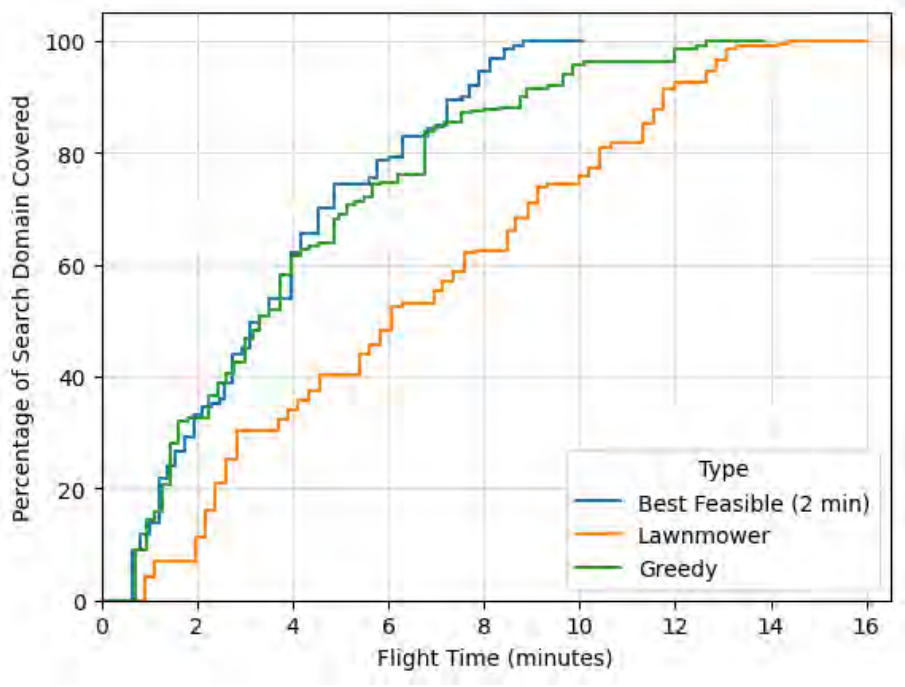
We record the UAV's odometry data then compare with our preplanned mission to determine the actual pose obtained at each vantage point. For each pre-planned vantage point, we determine the first time the UAV enters a radius of 1.5 m around it, then determine a target time for the UAV to be stationary within that radius, which is three seconds after entering. This wait time is used as a first filter, before filtering again to ensure yaws match the planned vantage point orientation within a small threshold. After these two filters, we choose the pose with minimal roll and pitch to ensure the UAV is level when performing inspection. At each of these vantage points, we use the UAV's full pose to find the ground plane intersection points for all pixels in the image to calculate the coverage obtained.

5.3.2 Temporal Analysis of Coverage

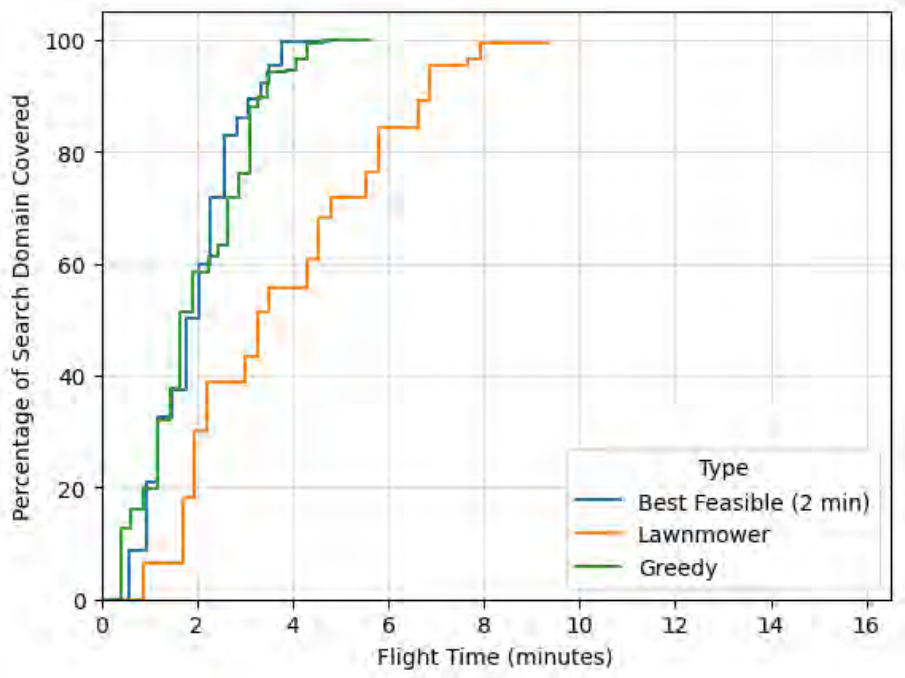
We start out by analyzing the effect of vantage point selection method on coverage over time. In simulation, we flew 6 missions, varying the spatial resolution threshold and the mission type. Figure 5.13 shows the percentage of the search domain covered over time for each of these missions. The most notable aspect from this plot is how at a sparser spatial resolution threshold, we are able to cover the search domain quicker for each mission type than their corresponding flight at a denser spatial resolution. When extracting the time from the first vantage point, which acts as the depot, until the UAV returns to that vantage point, at a threshold of 5 cm/pixel, the CP-SAT-based path plan took just over 9 minutes, the greedy path plan took just under 13 minutes, and the lawnmower path plan took just over 14 minutes. However, at the sparser threshold of 10 cm/pixel, the CP-SAT-based path plan took just over 3 and a half minutes, the greedy path plan took just under 5 minutes, and the lawnmower path plan took just over 7 and a half minutes.

Holding the mission type constant, this result can be explained by Figure 5.11d. Since we choose the flight altitude that maximizes the area of the effective sensor footprint, 19.5 m for the 5 cm/px threshold and 46 m for the 10 cm/px threshold, we can see that one vantage point can sufficiently cover significantly more area for the sparser resolution threshold. When solving for the minimum cardinality set of vantage points to cover the search domain, since the number of cells is constant, a larger effective sensor footprint means that we need fewer vantage points to cover the entire search domain. In turn, this means less total time over the entire mission performing inspection at vantage points and less travel time since we have fewer vantage points to travel to, resulting in an overall shorter mission time.

The order of mission times by path plan type is independent of spatial resolution. The best-feasible path plan was consistently the fastest, followed by the greedy algorithm, then the lawnmower



(a) 5 cm/pixel



(b) 10 cm/pixel

Figure 5.13: Percentage of search domain covered over time for different spatial resolution thresholds and mission types.

baseline. This follows the expected behavior of these specific coverage path plans. Given the two minutes of solving time with CP-SAT, we consistently saw that these mission plans achieved a near optimal number of vantage points based on the value of the optimality gap. Meanwhile the greedy solver can only give an approximate solution to Equation (3.12), resulting in more vantage points being generated. For example, at the 5 cm/px threshold, CP-SAT successfully covered the domain using only 44 vantage points, while the greedy solver required 72. At the 10 cm/px threshold, this trend continued, with CP-SAT and the greedy algorithm generating 14 and 22 vantage points, respectively.

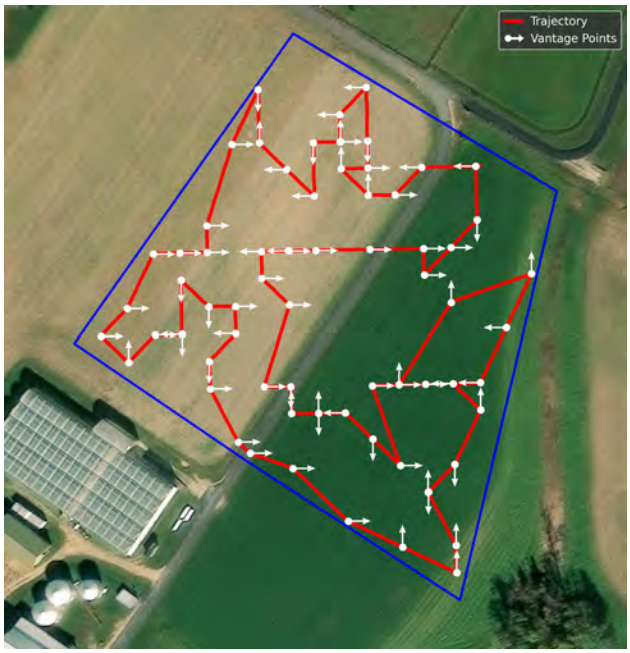
Interestingly though, for the 5 cm/px threshold, the lawnmower-derived pattern had 66 vantage points, which was less than the number of vantage points for the greedy path plan. However, the lawnmower pattern was still over a minute slower. Figure 5.14 compares these two flights. Upon closer examination of the vantage points, we can see that many of the greedy vantage points are overlapping. That is, multiple vantage points have the same (x, y, z) position but have a different yaw orientation, meaning there is no additional travel time between some of these vantage points. In addition, since the lawnmower pattern is not meant to be a cycle, the UAV has to fly all the way across the search domain to return to the depot.

5.3.3 Characterization of Coverage

Coverage is not just a binary metric though as there are different ways to characterize how vantage points cover a given region. The first is unique coverage, which denotes regions in the search domain that are covered by exactly one vantage point, sufficiently meeting the coverage resolution threshold. Overlapped coverage describes regions in the search domain that are covered by more than one vantage point, where each of these vantage points independently would cause cells in that region



(a) Lawnmower-derived



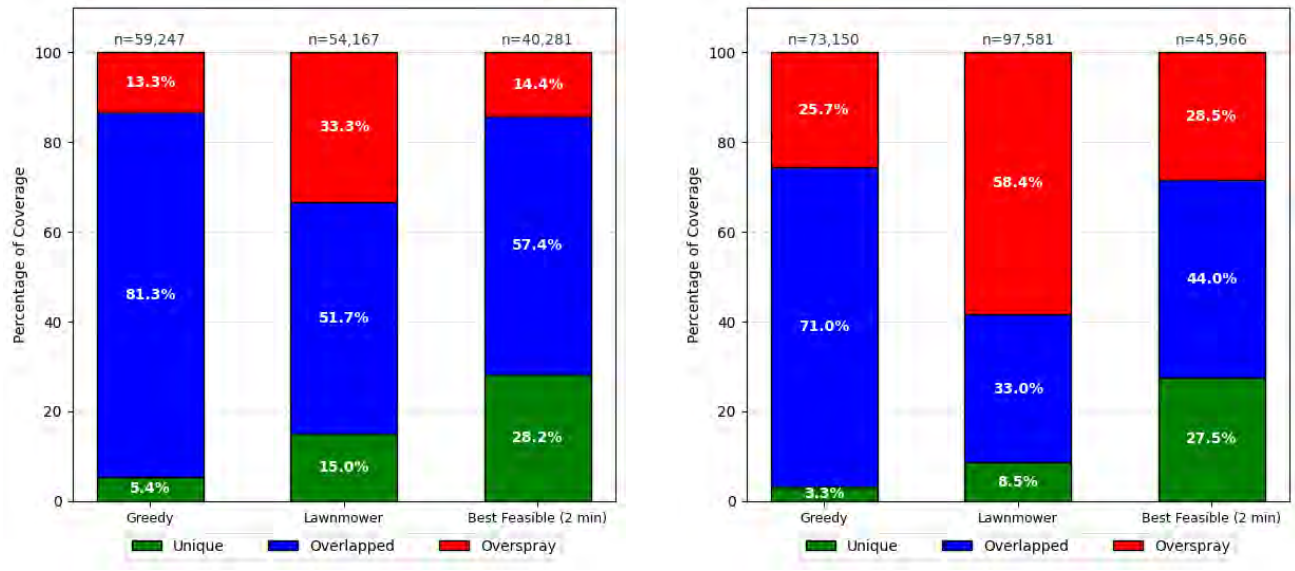
(b) Greedy

Figure 5.14: Vantage point coverage path plans flown in simulation at a spatial resolution of 5 cm/px and at an altitude of 19.5 m.

to be covered. Overspray coverage denotes regions outside of the search domain that unintentionally achieve the spatial resolution threshold. In an ideal world, an efficient coverage path plan would result in completely unique coverage, but this is not realistic. We will inevitably always have some overlap and overspray, but the goal with vantage point coverage path planning is to minimize these. While overspray is completely worthless to us, there is still some value to overlapped coverage. To analyze the characterization of coverage, we use the same six flights from Section 5.3.1.

Figure 5.15 shows the characterization of coverage for these six flights from simulation as a percentage of the total coverage viewed. The number at the top of each bar denotes the number of cell instances in the cumulative effective sensor footprint between all vantage points. For example, if a cell is covered by five vantage points independently, it counts for five cell instances. We can see that at each threshold, the lawnmower path plan results in the highest amount of overspray coverage, highlighting

this paradigm's inefficiencies. In fact, the proportion of overspray coverage is more than double that of the other two path planning methods at each threshold. This overspray is inherent to the lawnmower's structure as some sweeps will be close to and parallel to an edge of the search domain, resulting in wasted coverage at each vantage point since about half of the sensor footprint will be forced outside of the search domain. This wasted coverage is an unavoidable byproduct of lawnmower patterns, as there is no other way for the algorithm to guarantee complete coverage without over-sampling.



(a) 5 cm/px

(b) 10 cm/px

Figure 5.15: Characterization of vantage point coverage. n is the number of cell instances covered by vantage points.

On the other hand, the greedy path plan results in the highest proportion of overlapped coverage due to its inefficiency in solving the set cover problem. When examining the marginal gain in coverage for each successive vantage point selected, we often saw that many of the last remaining vantage points might cover fewer than 10 cells. Since the size of the effective sensor footprint is constant between vantage points, utilizing an entire footprint to capture such a small number of unique cells inherently wastes hundreds or thousands of cells of coverage on redundant overlapped or overspray coverage.

In contrast, CP-SAT minimizes the total number of vantage points selected, decreasing the amount of overlapping coverage which results in the highest proportion of unique coverage. Fewer vantage points in the same size search domain means that there is less possibility for overlap. Because the algorithm guarantees that every cell achieves the spatial resolution threshold during planning, any reduction in overlapping coverage inherently forces an increase in unique coverage.

5.3.4 Time-Limited Sequencing

While previous sections analyzed the accumulation of coverage over time for different coverage path planning methods, real-world operations often have temporal constraints that limit the amount of time available to search a region of interest, making it impossible to completely cover the entire area. This section evaluates different vantage point routing methods based on their ability to maximize the portion of the search domain covered within a time limit. In particular, we evaluate the performance of the Coverage Orienteering Problem.

In contrast to the previous missions, which aimed at complete coverage, we evaluate a new set of coverage path plans in a new search domain. We flew nine flights in simulation, all at a spatial resolution of 5 cm/px and an altitude of 46 m, varying the maximum mission time and routing method. The three time limits were 3, 5, and 10 minutes, where the time only consists of the actual search mission and not transit to search domain or transit after search. The routing methods used were a lawnmower-derived benchmark, Coverage OP, and the normal OP with the objective to maximize the number of vantage points. For the two OP variants, we formulated these problems in CP-SAT then took the best feasible solution after two minutes of solving time. The vantage points used for routing were generated using CP-SAT to solve Equation (3.12), also with a two minute solving limit. A trapezoidal



Figure 5.16: Search domain for time-limited flights in simulation.

velocity profile was used to estimate travel times, using an acceleration of 2.5 m/s^2 and a desired velocity of 5 m/s. For our lawnmower-derived benchmark, we chose track spacing heuristically to maximize the number of vantage points inside of the search domain within the time limit. The spacing between stops was chosen to be the height of the effective sensor footprint. Figure 5.16 shows the search domain we flew in, which was over 9 acres in size. All other procedures are the same as defined in Section 5.3.1.

Figure 5.17 shows how different routing methods affect the amount of the search domain that can be covered within each time limit. For all three time limits, the Coverage OP resulted in the greatest portion of the search domain covered, notably higher than the other two methods. Furthermore, the normal OP demonstrated a measurable, albeit smaller, improvement in total coverage compared to the lawnmower benchmark. For a maximum mission time of three minutes, the Coverage OP achieved 46.2% coverage, the normal OP achieved 41.9% coverage, and the lawnmower benchmark

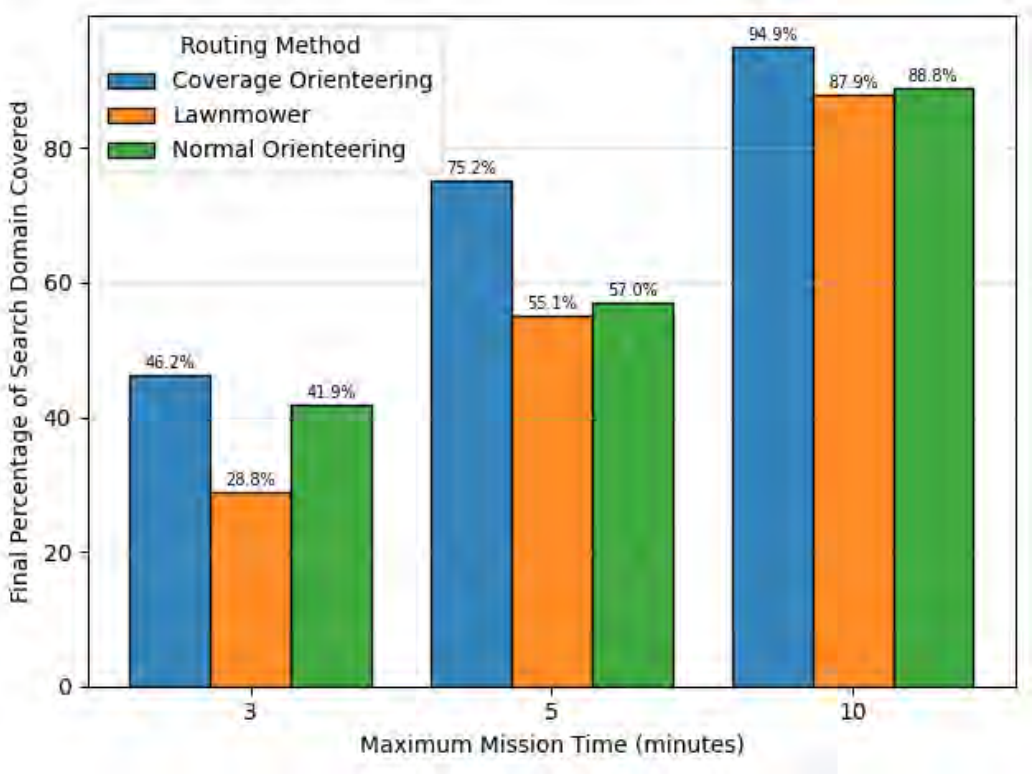
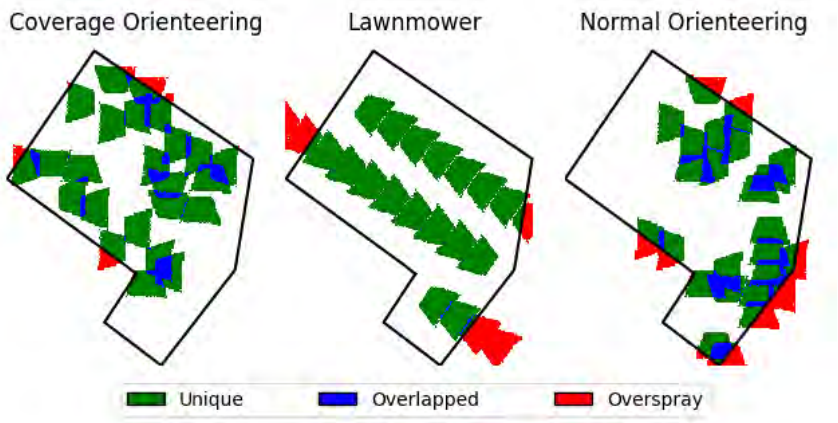


Figure 5.17: Percentage of search domain covered within time limit.

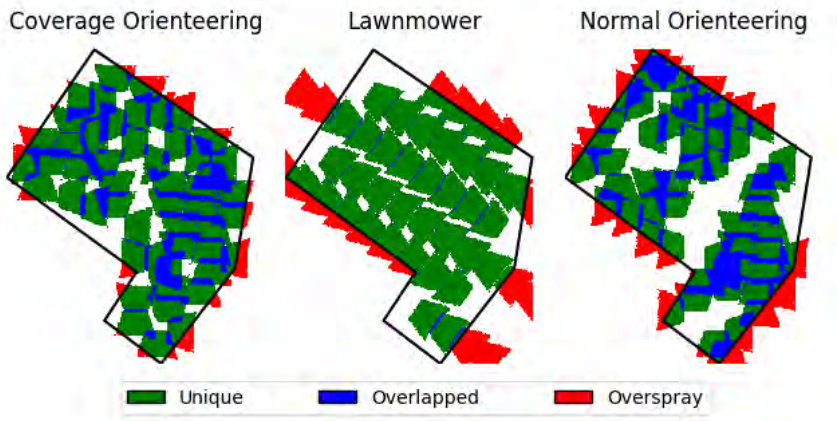
achieved 28.8% coverage. The difference between the Coverage OP and the other two routing methods increased with the five minute time limit as the Coverage OP achieved 75.2% coverage, while the normal OP and lawnmower achieved 57.0% and 55.1% coverage respectively. The gap in coverage between the three methods was the smallest at a time limit of ten minutes, as the Coverage OP achieved 94.9% coverage, the normal OP achieved 88.8% coverage, and the lawnmower path plan achieved 87.9% coverage.

Looking at Figure 5.18, we can examine how these methods choose vantage points and how that affects coverage. Starting with the lawnmower patterns, we can clearly see that the linear tracks result in pretty much only unique coverage inside of the search domain with practically no overlap, but does result in notable overspray outside of the search domain. It is worth noting that the amount of coverage that the lawnmower pattern can achieve is hindered due to the time traveling between

tracks. Meanwhile, the normal OP appears to focus on clustering vantage points, focusing on selecting vantage points that are very close to each other. The result is that while a high number of vantage points will be reached, there will also be a lot of overlapping coverage. Based on Table 5.1, we can see for each time limit, the normal OP travels to the highest number of vantage points. To contrast, the Coverage OP spreads out the selection of vantage points to try and maximize unique coverage.



(a) 5 minutes



(b) 10 minutes

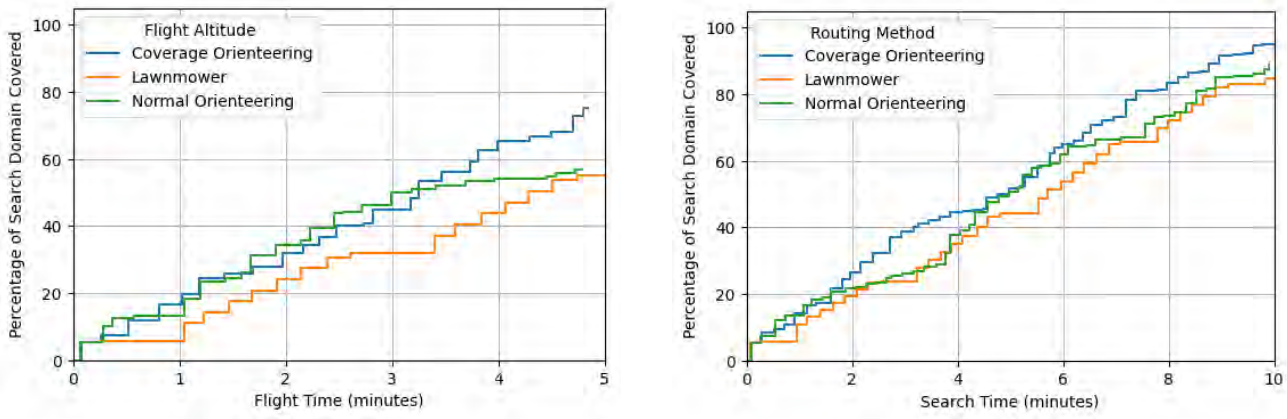
Figure 5.18: Characterization of coverage for different routing methods and time limits at a spatial resolution of 5 cm/px.

The effect of the normal OP focusing on vantage points that are close to each other can also be seen in Figure 5.19, which shows the percentage of the search domain covered over time for the three

	Time Limit (min)		
Routing Method	3	5	10
Coverage OP	13	24	51
Normal OP	15	25	55
Lawnmower	13	23	47

Table 5.1: Total number of vantage points visited by each routing method within time limit.

routing methods. Based on this figure, the normal OP will have stretches where the amount of the search domain covered sees only a marginal increase over time. This occurs when there is a cluster of vantage points in close proximity, resulting in lots of overlapping coverage but very little new unique coverage. The lawnmower pattern will also have stretches of no additional coverage, which is due to the travel between sweeps. These changes in sweep direction also directly follow vantage points that face outwards of the search domain and only achieve overspray. On the other hand, the Coverage OP sees consistent gains in coverage over time due to its selection of vantage points that will add lots of unique coverage, which is a result of the objective to maximize the number of cells covered.



(a) 5 minutes

(b) 10 minutes

Figure 5.19: Coverage over time for different routing methods at different time limits and a spatial resolution threshold of 5 cm/px.

Note that unlike the OP variants, which explicitly plan for a return to the depot, the lawnmower

plan is not a closed tour and still needs to come back to the depot after the mission is complete. In mission planners such as QGC, the tool to create lawnmower plans also does not create a closed loop, requiring an additional set of waypoints after the search mission is finished to bring the drone back to its starting point. In effect, the actual time for search is less for lawnmower patterns when budgeting the time for all aspects of flight. Because this subsequent return transit was not deducted from the lawnmower's temporal limit in our simulation, the baseline was effectively granted a larger operational budget for active search. Despite this temporal advantage, the lawnmower pattern still achieved the least amount of coverage over all three time limits, indicating that enforcing a strict depot-return constraint on this baseline would further widen the observed performance gap. These results show that with a limited time budget, the Coverage OP results in the highest portion of the search domain covered.

5.4 Experimental Results

To verify the operational validity of the proposed vantage point coverage path planning algorithm, real-world flight tests were performed. We flew six flights with a spatial resolution threshold of 10 cm/px at altitudes of 15 m, 30 m, and 45 m, evaluating the performance of the greedy solver versus the best feasible solution from CP-SAT after two minutes of solving time. All other setup is the same as Section 5.3.1. Flight testing was performed over two days separated by over a week in early February 2026. On the first day, the two missions at an altitude of 15 m were flown and the second day of flight testing completed the other four missions. It is important to note that flight testing was conducted during winds of over 13 knots.

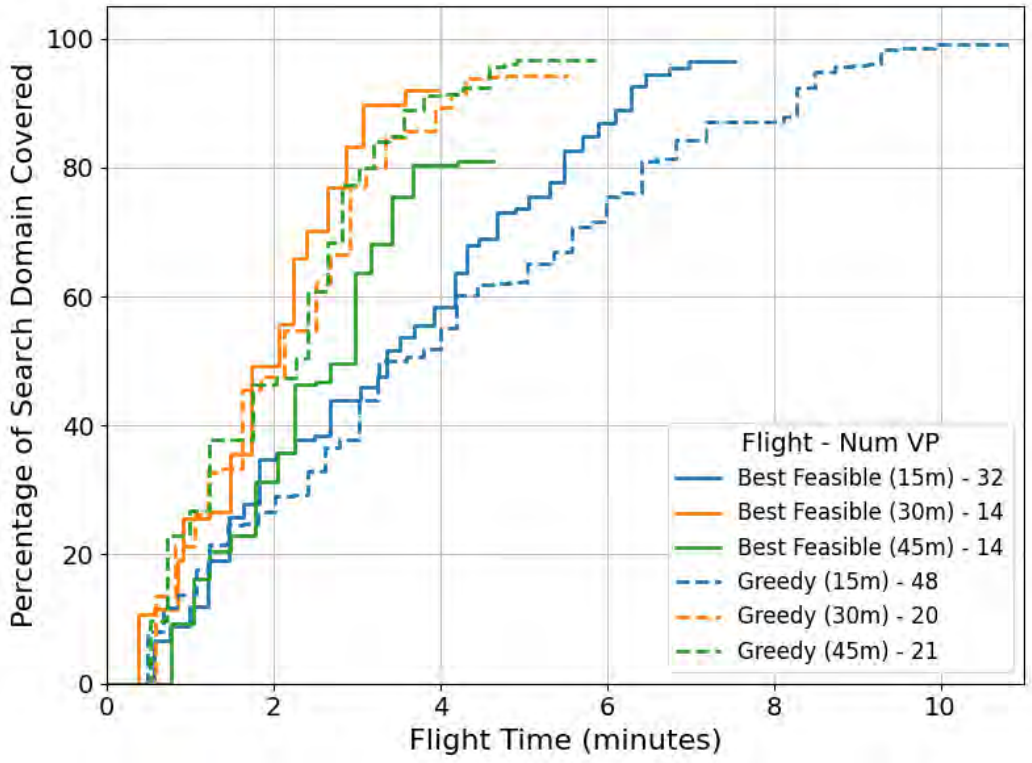
Figure 5.20 shows the coverage statistics from these flights. Unlike the non-time-limited simulation

tests, these flights did not achieve full coverage of the search domain. In some cases significant portions of the search domain were uncovered, such as 8.1% and 19.1% for the CP-SAT derived path plans at altitudes of 30 m and 45 m respectively. In fact, at each altitude, the greedy based path plan actually achieves a larger portion of coverage than the CP-SAT-derived path plan. However, at each altitude, all of the other coverage statistics go in favor of the CP-SAT-derived path plan because that flight achieves more unique coverage, less overlapped coverage, less overspray, and completes its mission in less time than the greedy path plan.

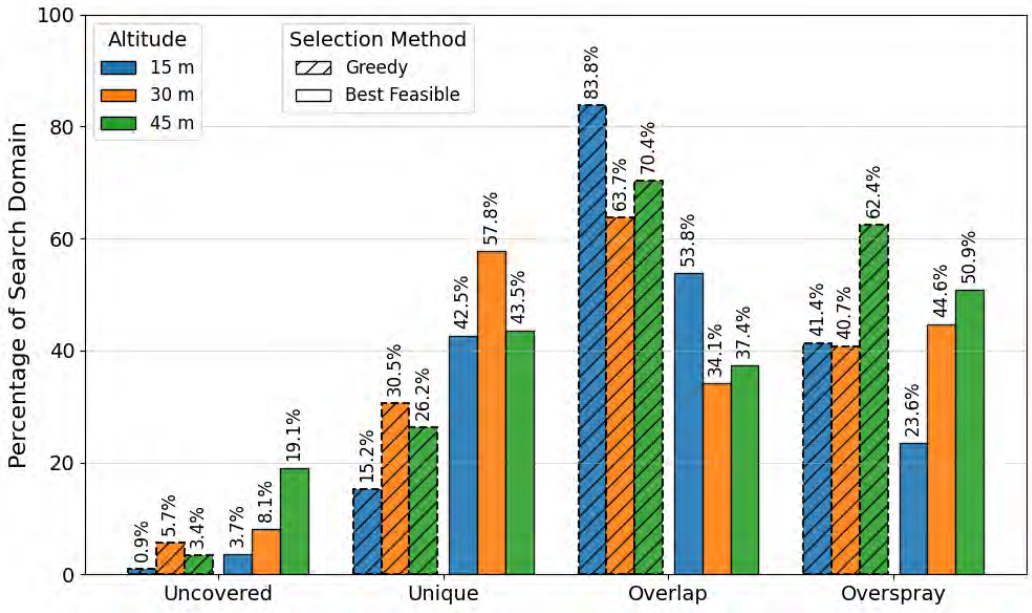
The default flight behavior of the Grazer-A prioritizes holding (x,y,z) position over maintaining a level attitude. When there is heavy wind, the flight controller will command heavy roll and pitch angles to fight against the wind, causing a discrepancy between the planned pose of the camera at each vantage point and the actual pose obtained. Since the Grazer-A does not have a gimbal, the attitude of the UAV directly affects the attitude of its onboard camera. As a result of this induced rotation, the size and shape of the effective sensor footprint can deviate drastically from their planned dimensions.

Figure 5.21 shows the realized effective sensor footprint at each vantage point for all six flights. This figure also includes a baseline effective sensor footprint to easily compare with, which is the effective sensor footprint at the given search altitude and without the UAV's roll or pitch. Note this baseline still accounts for the pitch from the camera relative to the UAV. It is very easy to see that the size and shape of each actual effective sensor footprint rarely match the dimensions of the baseline, experiencing severe geometric distortion and skewing.

This plot can also help explain why the greedy path plans result in a larger portion of the search domain covered, even though on the planning side we guarantee the entire search domain to be covered. The greedy path plans are not immune to these geometric distortion effects, but due to the higher number of vantage points, more of the search domain is able to be covered. In this case, the overlapping



(a) Coverage over time. Num VP is the number of vantage points in each path plan.



(b) Characterization of vantage point coverage relative to number of cells in search domain.

Figure 5.20: Real-world flight testing results with a spatial resolution threshold of 10 cm/px.

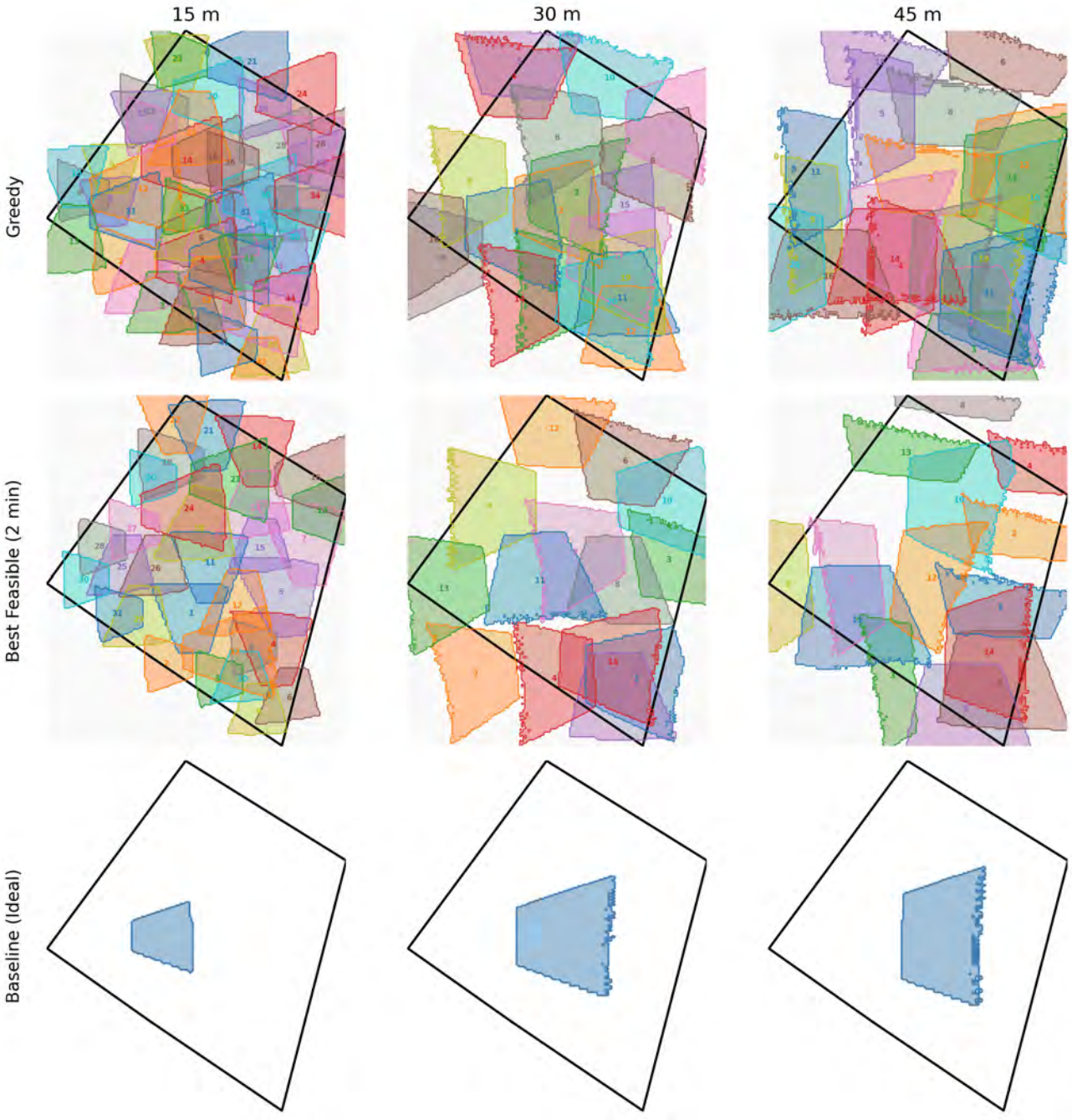
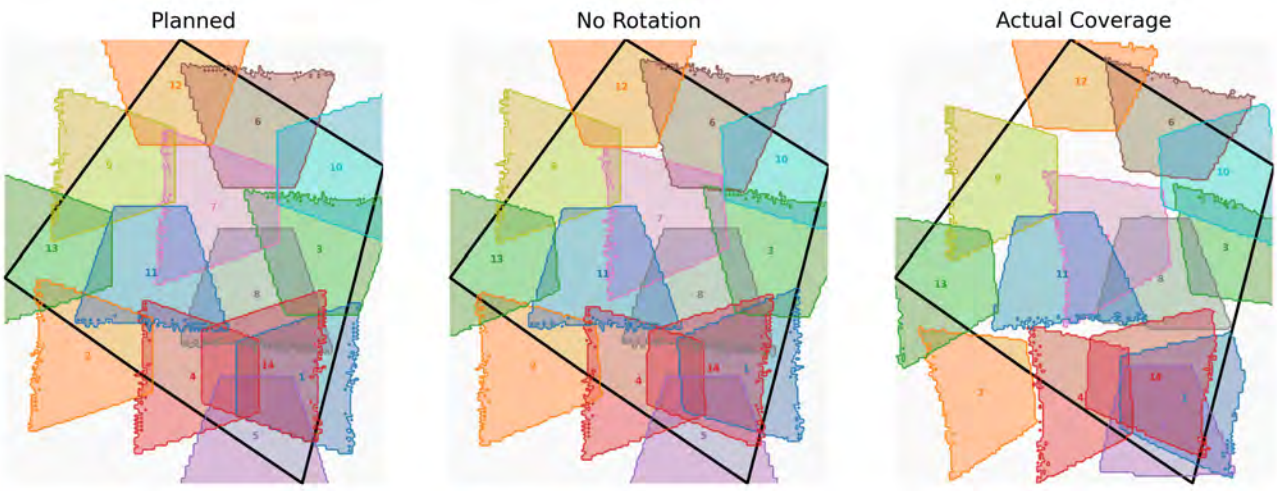


Figure 5.21: Effective sensor footprints at actual obtained vantage points from real-world testing at a spatial resolution of 10 cm/px.

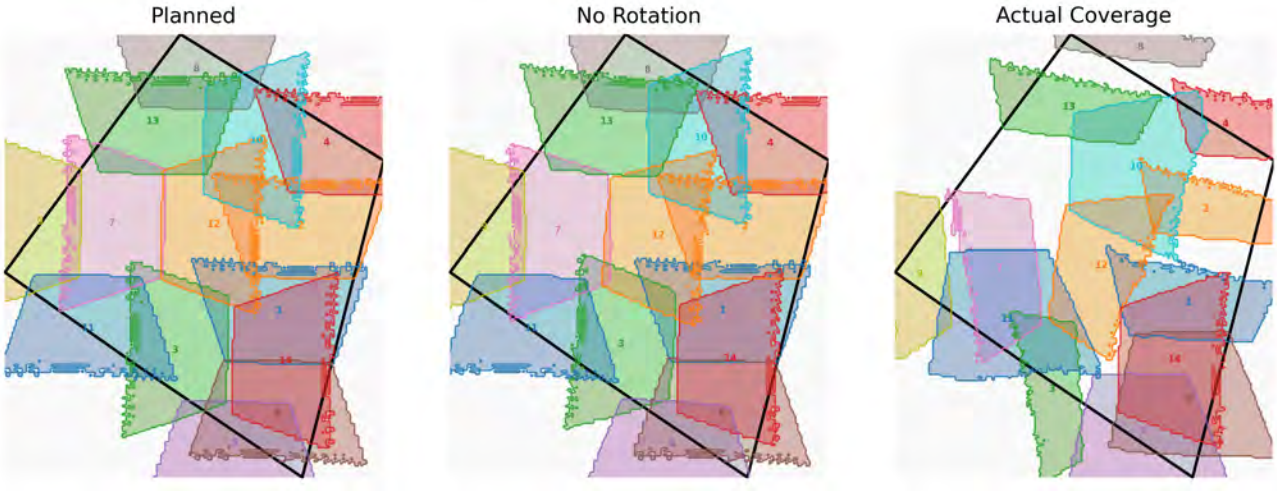
coverage serves as a buffer minimizing the amount of the search domain that is uncovered due to deviations from baseline UAV attitude. On the other hand, the CP-SAT-derived path plan does a much better job of minimizing overlap, so a large percentage of the search domain receives unique coverage. Thus, the coverage path plan is more vulnerable to geometric distortion of the effective sensor footprint since there is no margin for error in regions of unique coverage.

While there is some discrepancy between the position and yaw of the planned vantage point and the achieved vantage point, the difference in coverage is primarily due to the drone's roll and pitch. Figure 5.22 shows the map of the effective sensor footprints of the planned vantage points, the vantage points achieved during flight ignoring the effects of roll and pitch, and the actual effective sensor footprints. Without accounting for roll and pitch, the planned coverage and actual coverage remain largely the same. For example, in the 30 m altitude case in Figure 5.22a there is a gap in the actual coverage obtained without roll or pitch, but it is very minor in size. The significant coverage gaps arise once we do account for all axes of rotation in the calculation of coverage. Looking at Figure 5.23, we can see that if we ignore the UAV's pitch and roll then coverage estimates are much closer to what is planned. In each of the six flights less than 1% of the search domain does not meet the spatial resolution threshold.

The tradeoff between the speed of a coverage path plan and its robustness to degradation in flight is directly tied to the balance of unique and overlapping coverage. With minimal wind conditions, an increase in the proportion of unique coverage corresponds with a more efficient and quicker coverage path plan. With wind conditions, this type of idealized coverage path plan is not robust enough to ensure full coverage. This does not mean that a lawnmower pattern is the answer since it would be susceptible to the same flaws — the solution is better modeling of the effective sensor footprint and its uncertainty, not blindly planning for overlap.

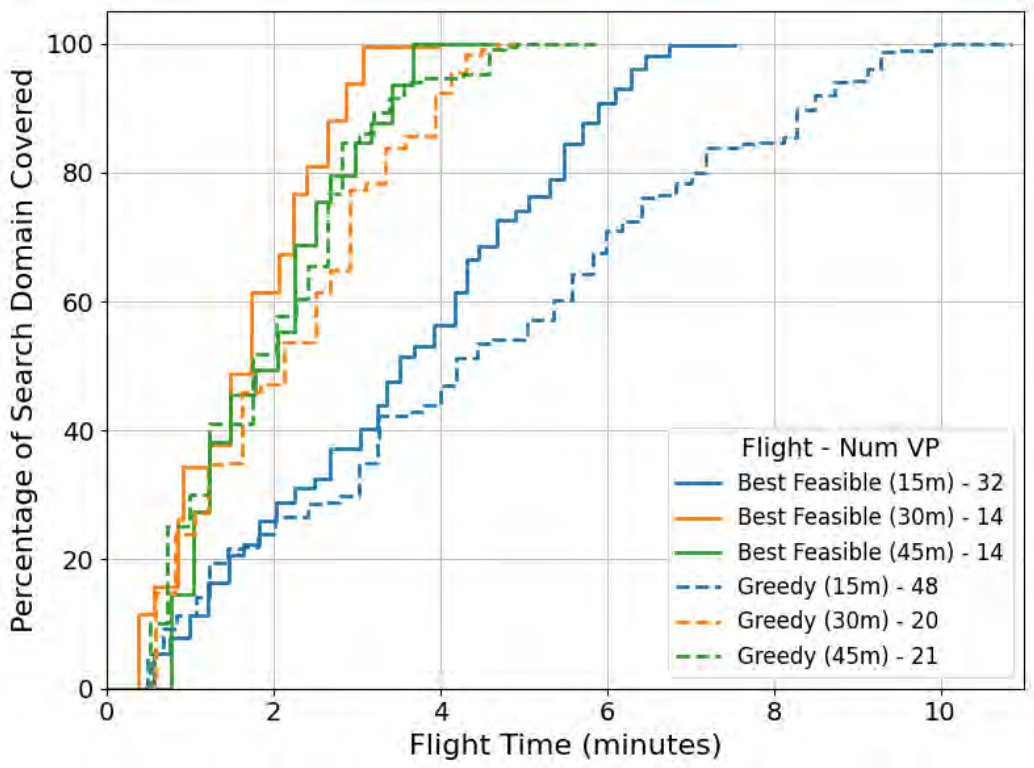


(a) 30 m altitude

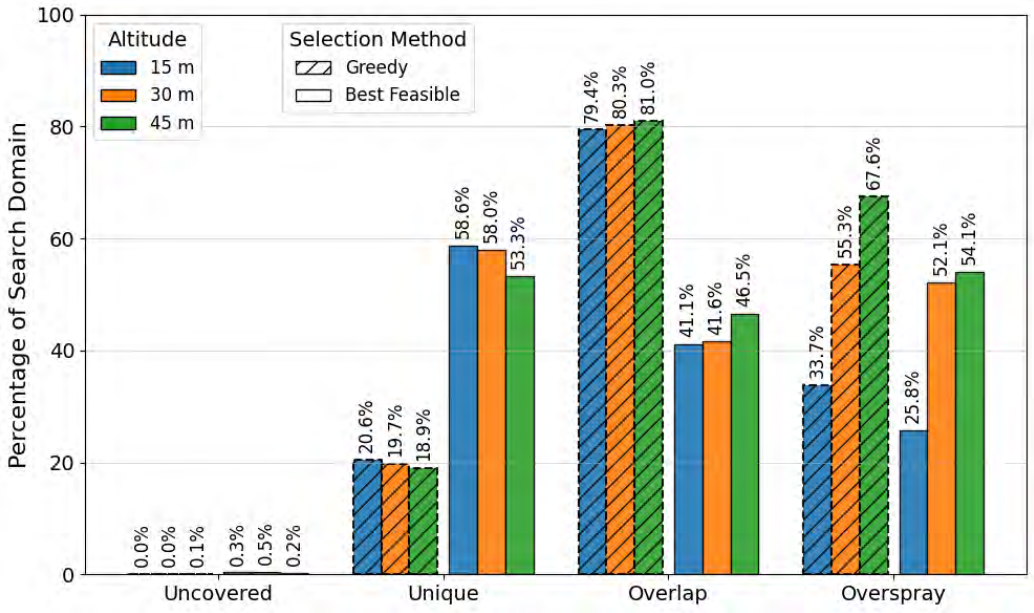


(b) 45 m altitude

Figure 5.22: Coverage maps using different levels of vantage point fidelity from real-world flight testing at a spatial resolution of 10 cm/px for the best feasible solution of 2 minutes of solving time from CP-SAT.



(a) Coverage over time. Num VP is the number of vantage points in each path plan.



(b) Characterization of vantage point coverage relative to number of cells in search domain.

Figure 5.23: Real-world flight testing results with a spatial resolution threshold of 10 cm/px, ignoring UAV pitch and roll.

Chapter 6: Conclusion

6.1 Summary of Contributions

This thesis proposes a new method for UAV coverage path planning that provides more efficient path plans compared to conventional methods. This coverage path plan is defined by a sequence of poses called vantage points that the UAV will be stationary at to perform inspection. First, this thesis describes a method for generating a set of vantage points that guarantee the entire search domain achieves a spatial resolution threshold. At the core of this algorithm is the formulation of an integer linear program that solves for the minimum cardinality set of vantage points meeting this coverage requirement, in effect minimizing the amount of coverage wasted outside the search domain and wasted on previously covered regions. Second, when faced with temporal constraints in the form of a maximum mission time, this thesis formulates a new vehicle routing problem variant that maximizes the area of the search domain that meets the spatial resolution threshold within the time limit. Finally, this thesis provides experimental results to evaluate the coverage path plans generated by this method, comparing with a lawnmower-derived baseline. Data from simulation shows that this method results in quicker and more efficient coverage. However, real-world flight testing reveals that the quality of coverage degrades due to wind-induced attitude disturbances. Overall, while further development for robustness to environmental effects is needed for actual deployment for a UAV without a gimbal, this thesis establishes a constraint optimization-based framework for efficient UAV coverage path planning.

6.2 Suggestions For Future Work

To increase the applicability of vantage point coverage path planning for real-world use, future work should focus on the following areas:

- **Dynamic Search**: Section 5.4 showed that there can be degradation in the quality of coverage between the planned coverage path plan and what is actually flown. Based on the actual coverage obtained in flight, the coverage path plan should adapt the set of vantage points to fix any potential gaps in coverage. In addition, to enhance real-time planning capabilities, vantage point routing should be performed online in order to increase the computational time available for solving VRPs. As the UAV flies, the sequence of vantage points should be continually updated based on the current best feasible solution to the VRP and also the current coverage obtained.

- **In-Flight Stability**: Since the coverage from a given vantage point can degrade significantly with deviations of the camera from level attitude during flight, such as when trying to hold a setpoint with wind, effort should be spent trying to mitigate the effects of this. The simplest possible solution is using a gimbal to keep the camera level and stabilized. One key point of note is that the gimbal should ensure that the pitch of the camera is the same as the preplanned mission with respect to the world frame instead of the UAV body frame. If it is not possible to use a gimbal, one potential solution is to change the behavior of the controller at vantage points to prioritize attitude stabilization over positional accuracy.

- **Vantage Point Generation**: The two methods described in this thesis for solving Equation (3.12) to generate vantage points are two extremes of ILP solving methods. Exploring approximation algorithms that can mitigate the tradeoff between solving time and solution accuracy would greatly improve this coverage path planning algorithm's usage in real-world environments. Similar to dynamic search, another approach is to only generate vantage points for regions of the search domain in proximity to the UAV and keep generating vantage points online throughout flight. This method of dynamic vantage point generation would also improve solving times for larger

search domains.

- **Uneven Terrain Altitude**: While the flat Earth assumption can be a good approximation for many environments, such as open fields, if the terrain has uneven elevation then a more accurate method must be used for calculating the effective sensor footprint at a vantage point. For a full fidelity estimation, using a digital elevation model, which describes the altitude above sea level for each region on the ground, can allow for a more accurate representation of the point in the world frame corresponding to each pixel in the image. One important thing of note is that in order to calculate this intersection point, since we no longer assume the altitude of the ground plane intersection is zero, we have to perform ray casting. This method can also easily be extended to account for occlusions in the environment such as trees or buildings.

Appendix A: Calibration Parameters for Grazer-A Hi-Res Camera

Table A.1: Camera Intrinsic and Distortion Parameters

Parameter	**Value**
Camera Name	Grazer Hi-Res
Image Resolution	1280×768 px
Distortion Model	Plumb Bob

Camera Matrix $\mathbf{K}$

$$\mathbf{K} = \begin{bmatrix} 1498.109815 & 0 & 639.500000 \\ 0 & 1498.109815 & 383.500000 \\ 0 & 0 & 1 \end{bmatrix}$$

Distortion Coefficients

$$\mathbf{d} = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix} = \begin{bmatrix} 0.124517 & 0.114196 & -0.010649 & 0.026845 & 0.000000 \end{bmatrix}$$

Bibliography

[1] W. Cui, "Applied aerial robotics for long range autonomy and advanced perception," Ph.D. dissertation, University of Maryland, College Park, 2024.

[2] R. S. Kotwal, J. T. Howard, J. A. Orman, B. W. Tarpey, J. A. Bailey, H. R. Champion, R. L. Mabry, J. B. Holcomb, and K. R. Gross, "The effect of a golden hour policy on the morbidity and mortality of combat casualties," *JAMA surgery*, vol. 151, no. 1, pp. 15–24, 2016.

[3] M. Lyu, Y. Zhao, C. Huang, and H. Huang, "Unmanned aerial vehicles for search and rescue: A survey," *Remote Sensing*, vol. 15, no. 13, 2023. [Online]. Available: https://www.mdpi.com/2072-4292/15/13/3266

[4] S. M. S. Mohd Daud, M. Y. P. Mohd Yusof, C. C. Heo, L. S. Khoo, M. K. Chainchel Singh, M. S. Mahmood, and H. Nawawi, "Applications of drone in disaster management: A scoping review," *Science Justice*, vol. 62, no. 1, pp. 30–42, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1355030621001477

[5] A. Otto, N. Agatz, J. Campbell, B. Golden, and E. Pesch, "Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey," *Networks*, vol. 72, no. 4, pp. 411–458, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.21818

[6] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Field and service robotics*. Springer, 1998, pp. 203–209.

[7] T. M. Cabreira, L. B. Brisolara, and P. R. Ferreira Jr., "Survey on coverage path planning with unmanned aerial vehicles," *Drones*, vol. 3, no. 1, 2019. [Online]. Available: https://www.mdpi.com/2504-446X/3/1/4

[8] Dronecode Project, Inc., "Qgroundcontrol - drone control," 2025. [Online]. Available: https://qgroundcontrol.com/

[9] Z. Bortoff, A. Luterman, D. A. Paley, and S. M. Nogar, "Using Target Detection Probability to Evaluate Area Coverage by a UAV," in *2025 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2025, pp. 572–578.

[10] M. Yazdi and T. Bouwmans, "New trends on moving object detection in video images captured by a moving camera: A survey," *Computer Science Review*, vol. 28, pp. 157–177, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013716301794

[11] D. Kingston, S. Rasmussen, and L. Humphrey, "Automated uav tasks for search and surveillance," in *2016 IEEE Conference on Control Applications (CCA)*, 2016, pp. 1–8.

[12] J. C. Leachtenauer and R. G. Driggers, *Surveillance and reconnaissance imaging systems: modeling and performance prediction*. Artech House, 2001.

[13] J. Zhang and H. Huang, "Occlusion-aware uav path planning for reconnaissance and surveillance," *Drones*, vol. 5, no. 3, p. 98, 2021.

[14] 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, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/6/1438

[15] J. O'rourke, *Art gallery theorems and algorithms*. Oxford University Press, Inc., 1987.

[16] M. Marengoni, B. Draper, A. Hanson, and R. Sitaraman, "A system to place observers on a polyhedral terrain in polynomial time," *Image and Vision Computing*, vol. 18, no. 10, pp. 773–780, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0262885699000451

[17] P. A. Plonski and V. Isler, "Approximation algorithms for tours of height-varying view cones," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 224–235, 2019. [Online]. Available: https://doi.org/10.1177/0278364918784353

[18] N. Stefas, P. A. Plonski, and V. Isler, "Approximation algorithms for tours of orientation-varying view cones," *The International Journal of Robotics Research*, vol. 39, no. 4, pp. 389–401, 2020.

[19] P. Tokekar, A. K. Budhiraja, and V. Kumar, "Algorithms for visibility-based monitoring with robot teams," *arXiv preprint arXiv:1612.03246*, 2016.

[20] H. Dhami, K. Yu, T. Williams, V. Vajipey, and P. Tokekar, "GATSBI: An online gtsp-based algorithm for targeted surface bridge inspection," in *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2023, pp. 1199–1206.

[21] B. Bogaerts, S. Sels, S. Vanlanduit, and R. Penne, "Near-optimal path planning for complex robotic inspection tasks," *arXiv preprint arXiv:1905.05528*, 2019.

[22] A. Luterman, Z. Bortoff, D. A. Paley, and S. M. Nogar, "Time-constrained coverage path planning for UAV search applications," in *2026 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2026, to appear.

[23] J. Nocedal and S. J. Wright, *Numerical optimization*, 2nd ed. Springer, 2006.

[24] S. S. Rao, *Engineering optimization: theory and practice*. John Wiley & Sons, 2019.

[25] D. P. Williamson and D. B. Shmoys, *The design of approximation algorithms*. Cambridge university press, 2011.

[26] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[27] L. Perron and V. Furnon, "Or-tools," Google, v9.12. [Online]. Available: https://developers.google.com/optimization/

[28] V. Furnon and L. Perron, "Or-tools routing library," Google, v9.12. [Online]. Available: https://developers.google.com/optimization/routing/

[29] MiniZinc. The minizinc challenge. [Online]. Available: https://www.minizinc.org/challenge/

[30] K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse, "The vehicle routing problem: State of the art classification and review," *Computers Industrial Engineering*, vol. 99, pp. 300–313, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835215004775

[31] N. Christofides, "The vehicle routing problem," *Revue française d'automatique, informatique, recherche opérationnelle. Recherche opérationnelle*, vol. 10, no. V1, pp. 55–70, 1976.

[32] B. L. Golden, S. Raghavan, and E. A. Wasil, *The Vehicle Routing Problem: Latest Advances and New Challenges*, ser. Operations Research/Computer Science Interfaces Series. Springer US, 2008. [Online]. Available: https://books.google.com/books?id=-3ta5ne3-owC

[33] G. Laporte, "What you should know about the vehicle routing problem," *Naval Research Logistics (NRL)*, vol. 54, no. 8, pp. 811–819, 2007. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.20261

[34] G. Gutin and A. P. Punnen, *The traveling salesman problem and its variations*. Springer Science & Business Media, 2006, vol. 12.

[35] N. Christofides, "Technical note—bounds for the travelling-salesman problem," *Operations Research*, vol. 20, no. 5, pp. 1044–1056, 1972. [Online]. Available: https://doi.org/10.1287/opre.20.5.1044

[36] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *Journal of the ACM (JACM)*, vol. 7, no. 4, pp. 326–329, 1960.

[37] T. Bektaş and L. Gouveia, "Requiem for the miller–tucker–zemlin subtour elimination constraints?" *European Journal of Operational Research*, vol. 236, no. 3, pp. 820–832, 2014, vehicle Routing and Distribution Logistics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S037722171300619X

[38] D. Bienstock, M. X. Goemans, D. Simchi-Levi, and D. Williamson, "A note on the prize collecting traveling salesman problem," *Mathematical programming*, vol. 59, no. 1, pp. 413–420, 1993.

[39] B. L. Golden, L. Levy, and R. Vohra, "The orienteering problem," *Naval Research Logistics (NRL)*, vol. 34, no. 3, pp. 307–318, 1987.

[40] T. Tsiligirides, "Heuristic methods applied to orienteering," *The Journal of the Operational Research Society*, vol. 35, no. 9, pp. 797–809, 1984. [Online]. Available: http://www.jstor.org/stable/2582629

[41] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, no. 1, pp. 1–10, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221710002973

[42] A. Blum, S. Chawla, D. R. Karger, T. Lane, A. Meyerson, and M. Minkoff, "Approximation algorithms for orienteering and discounted-reward TSP," *SIAM Journal on Computing*, vol. 37, no. 2, pp. 653–670, 2007.

[43] D. Grundel and D. Jeffcoat, "Formulation and solution of the target visitation problem," in *AIAA 1st Intelligent Systems Technical Conference*, 2004, p. 6212.

[44] K. Panchamgam, Y. Xiong, B. Golden, B. Dussault, and E. Wasil, "The hierarchical traveling salesman problem," *Optimization Letters*, vol. 7, no. 7, pp. 1517–1524, Oct. 2013. [Online]. Available: https://doi.org/10.1007/s11590-012-0553-x

[45] W. Zhang, K. Wang, S. Wang, and G. Laporte, "Clustered coverage orienteering problem of unmanned surface vehicles for water sampling," *Naval Research Logistics (NRL)*, vol. 67, no. 5, pp. 353–367, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.21906

[46] D. Hall, F. Dayoub, J. Skinner, H. Zhang, D. Miller, P. Corke, G. Carneiro, A. Angelova, and N. Sünderhauf, "Probabilistic object detection: Definition and evaluation," in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020, pp. 1020–1029.

[47] T. Ali, A. Z. Sadeque, M. Saquib, and M. Ali, "Mimo radar for target detection and localization in sensor networks," *IEEE Systems Journal*, vol. 8, no. 1, pp. 75–82, 2014.

[48] A. Quazi, "An overview on the time delay estimate in active and passive systems for target localization," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 3, pp. 527–533, 1981.

[49] G. Jocher, J. Qiu, and A. Chaurasia, "Ultralytics YOLO," Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[50] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7310–7311.

[51] K. Ikeuchi, *Computer Vision: A Reference Guide*, ser. Computer Vision. Springer International Publishing, 2021. [Online]. Available: https://books.google.com/books?id=E0oEzgEACAAJ

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

[53] R. Szeliski, *Computer Vision: Algorithms and Applications*, ser. Texts in Computer Science. Springer International Publishing, 2022. [Online]. Available: https://szeliski.org/Book/

[54] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[55] P. Drap and J. Lefèvre, "An exact formula for calculating inverse radial lens distortions," *Sensors*, vol. 16, no. 6, 2016. [Online]. Available: https://www.mdpi.com/1424-8220/16/6/807

[56] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 1106–1112.

[57] T. Carchrae and J. C. Beck, "Principles for the design of large neighborhood search," *Journal of Mathematical Modelling and Algorithms*, vol. 8, no. 3, pp. 245–270, 2009.

[58] P. Shaw, "Using constraint programming and local search methods to solve vehicle routing problems," in *Principles and Practice of Constraint Programming — CP98*, M. Maher and J.-F. Puget, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 417–431.

[59] S. T. Windras Mara, R. Norcahyo, P. Jodiawan, L. Lusiantoro, and A. P. Rifai, "A survey of adaptive large neighborhood search algorithms and applications," *Computers Operations Research*, vol. 146, p. 105903, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054822001654

[60] W. Cui, A. Shastry, D. A. Paley, and S. Nogar, "Autonomous aerial search and revisit behavior for communication limited environments," in *AIAA SCITECH 2024 Forum*, 2024, p. 1170.

[61] Esri, "World imagery," 2026, accessed: February 23, 2026. [Online]. Available: https://www.arcgis.com/home/item.html?id=10df2279f9684e4a9f6a7f08febac2a9