Minimizing Turns in Single and Multi Robot Coverage Path Planning

by

Stanislav Bochkarev

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2017

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Spurred by declining costs of robotics, automation is becoming a prevalent area of interest for many industries. In some cases, it even makes economic sense to use a team of robots to achieve a goal faster. In this thesis we study sweep coverage path planning, in which a robot or a team of robots must cover all points in a workspace with its footprint. In many coverage applications, including cleaning and monitoring, it is beneficial to use coverage paths with minimal robot turns.

In the first part of the thesis, we address this for a single robot by providing an efficient method to compute the minimum altitude of a non-convex polygonal region, which captures the number of parallel line segments, and thus turns, needed to cover the region. Then, given a non-convex polygon, we provide a method to cut the polygon into two pieces that minimizes the sum of their altitudes. Given an initial convex decomposition of a workspace, we apply this method to iteratively re-optimize and delete cuts of the decomposition. Finally, we compute a coverage path of the workspace by placing parallel line segments in each region, and then computing a tour of the segments of minimum cost. We present simulation results on several workspaces with obstacles, which demonstrate improvements in both the number of turns in the final coverage path and runtime.

In the second part of the thesis, we extend the concepts developed for a single robot coverage to a multi robot case. We provide a metric χ that approximates the cost of a coverage path, which accounts for the cost of turns. Given a polygon, we provide a method for cutting a polygon into two that would minimize the maximum cost χ between the two polygons. Provided with an initial n-cell decomposition, we apply this method in the iterative manner to re-optimize cuts in order to minimize the maximum cost χ over all cells in the decomposition. For each cell in the re-optimized n-cell decomposition, a single robot coverage path is computed using the minimum altitude decomposition. We present the simulation results that demonstrate improvements in the maximum cost as well as the range of costs over all robots in the team.

Acknowledgments

I would like to thank my advisor Stephen Smith for his guidance and endless support during my Master's career. I would also like to thank Adam Gomes and Ryan McDonald for their help and numerous insightful discussions.

Table of Contents

List of Figures viii

List of Tables x

1 Introduction 1

1.1 Single and Multi-agent Coverage Path Planning with Minimum Turns . . . 1

1.2 Thesis Contribution . . . 3

1.3 Organization . . . 4

2 Literature Review 5

2.1 Exact Cellular Decomposition . . . 6

2.2 Approximate Decomposition or Grid Based Approaches . . . 8

2.3 Minimizing Turns in Coverage Paths . . . 9

2.4 Multi-agent Coverage . . . 9

3 Background 11

3.1 Robot Dynamics . . . 11

3.2 Geometric Preliminaries . . . 13

3.3 Traveling Salesman Problem Preliminaries . . . 14

4 Single Agent Coverage **18**
4.1 Problem Statement . . . 18
4.1.1 Utilizing Segmented Paths . . . 19
4.1.2 Path Evaluation Metric . . . 20
4.1.3 Subclass of Segmented Paths: Boustrophedon . . . 21
4.1.4 Coverage with Minimum Turns . . . 22
4.1.5 Decoupled Problem Statement . . . 23
4.2 Minimum Altitude Polygon Decomposition for Coverage Planning . . . 25
4.2.1 Minimum Altitude Decomposition . . . 25
4.2.2 Optimal Decomposing Cut . . . 26
4.2.3 Proof of Correctness . . . 26
4.3 Computational Complexity . . . 30
4.4 GTSP Tour Generation . . . 32
4.5 Simulations . . . 34

5 Multi-agent Coverage **37**
5.1 Problem Statement . . . 37
5.2 Decomposition Metric . . . 40
5.2.1 Initialization and Return Distance . . . 40
5.2.2 Length of Straight Line Segments . . . 42
5.2.3 Angular Component . . . 44
5.3 Area Allocation Algorithm . . . 46
5.4 Computational Complexity . . . 50
5.5 Simulations . . . 52

6 Conclusions and Future Work **58**
6.1 Future Work on Single Agent Coverage . . . 58
6.2 Improvements to Multi-Agent Coverage . . . 60

References **61**

A Uncovered Area Term Derivation **68**

List of Figures

1.1 An example of an inspection coverage performed by a robotic arm. 2
1.2 Coverage footprint over a line. 3

3.1 An example of a workspace with a disk robot. 12
3.2 An example of a free configuration space shown in orange for a point robot. 12
3.3 An example of a coverable space show in green. 13
3.4 Decomposing cut originating at a reflex vertex. 15
3.5 An example of a complete graph G. 15
3.6 An example of a TSP tour on a complete graph G. 16
3.7 An example of a GTSP tour on a complete graph G. 17

4.1 An example of a segmented path. 21
4.2 Parallel line arrangement. 21
4.3 Polygon regions where lines can be reoriented. 22
4.4 Process of measuring altitude with θ equal to 0. 24
4.5 An example of a pinch vertex at v_p. 28
4.6 An example of a transition point. 29
4.7 Three cases of altitude behavior for W_ℓ. 31
4.8 Three cases of altitude behavior for W_r. 31
4.9 Two possible traversal directions for line γ_j. 33
4.10 Process of converting a set of lines into a graph, followed by a GTSP instance. 33

4.11 First column: point decomposition. Second column: greedy decomposition. Third column: min altitude decomposition. 36

5.1 The structure of a coverage path. Orange: Segments where the robot travels to its assigned region shown in blue. Red: Path segments that are straight line segments. Yellow: path segments that are transition segments. 41

5.2 Coverage footprints over lines. 43

5.3 Minimum altitude of a convex polygon, α. Distance to the center contour, β. 46

5.4 Top: initial decomposition. Middle: after first iteration. Bottom: after second iteration. 50

5.5 Decomposition results for shape 1,2, and 3. First column: decomposition before re-optimization. Second column: after re-optimization. 55

5.6 Decomposition results for shape 4 and 5. First column: decomposition before re-optimization. Second column: after re-optimization. 56

5.7 Hole Re-optimization results for shape 4 and 5. First column: decomposition before re-optimization. Second column: after re-optimization. 57

6.1 An example of a undesired cell transition. 59

6.2 An example of a improvement to inter-cell transitions. 59

6.3 An example where the current re-optimization technique terminates. 60

A.1 Sample workspace with parallel line segments. 69

A.2 Geometry of the uncovered area. 70

List of Tables

4.1 Decomposition methods comparison for multiple test shapes. 35

5.1 Improvement to the maximum cost. 53

5.2 Improvement to the range of costs. 53

Chapter 1

Introduction

1.1 Single and Multi-agent Coverage Path Planning with Minimum Turns

The human reliance on robots to keep the society functioning is increasing. Enormous amounts of resources are invested in automation in robotics by companies looking to enter new markets, decrease operational costs, increase efficiency or solve problems that were impossible before. There are numerous applications that stand to benefit from increased degree of automation in robotics. A certain subset of these applications include highly important tasks such as search and rescue [56], natural disaster monitoring and relief [22], demining [3], and surveillance [53]. Other examples include tasks related to operations research such as floor sweeping [38], factory automated painting [58], crop health monitoring [57], and ship hull inspection [63]. Despite the differences between all these applications, these problems are related to a common problem in robotics called the coverage path planning problem.

Given a robot with a footprint $\mathcal{M}$ and a workspace W, possibly with holes, the coverage problem is that of computing a path contained in W such that traversal of $\mathcal{M}$ along the path results in each point in W being covered by $\mathcal{M}$. An example of a robot performing a coverage task is shown in Figure 1.1 where a robotic inspection arm inspects a test surface. The problem is as old as the machine controlled milling where operators were generating paths for the tool [33]. However, it is only in 2000 that Arkin [5] has demonstrated that the problem is in fact NP-complete. As such, there are numerous suboptimal solutions that have been proposed over the years. Moreover, there is an extension to this problem that

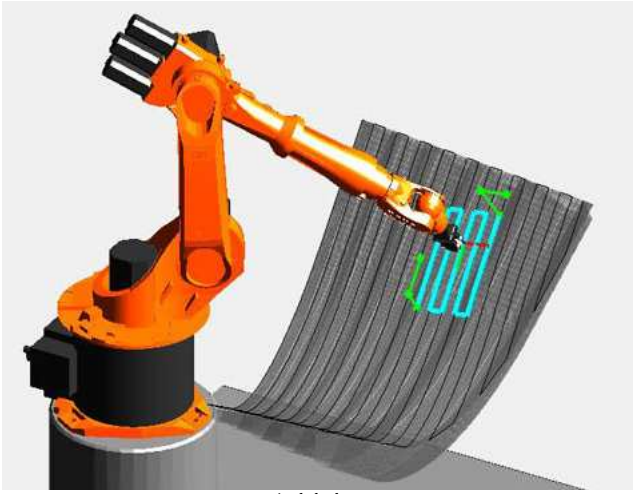

Figure 1.1: An example of an inspection coverage performed by a robotic arm.

has been gaining popularity over the recent years. Spurred by decreasing costs of robotics, the multi-robot systems for coverage are becoming more appealing.

It is difficult to discuss good solutions to the coverage path planning problem because the notion of *goodness* varies from one application to another. For example, a good coverage path for a painting robot would result in the most uniform paint layer on the surface. A good coverage path for a crop monitoring application would yield a high quality capture of the entire field. As such, works in literature that focus on one application offers solutions that are specialized to that application. On the other hand, numerous other works offer theoretical analysis and algorithmic approaches to the problem. These tend to propose solutions that theoretically achieve complete coverage; however, the solutions tend to posses undesirable properties such as excessive amount of turns that make their implementation in practice challenging.

In this work we aim to compute coverage paths with properties that make them desirable in practice by minimizing the amount of turns in the coverage path. Our key idea is to perform a line decomposition: an approximate decomposition of the workspace into regions that represent the area swept by the footprint traversing a straight line (Figure 1.2). The process starts with any convex decomposition. We propose a method of re-evaluating cuts in this convex decomposition with the objective of lowering the number of turns in the path. For each polygon in the final decomposition, a minimal line decomposition is performed. The complete coverage path is then generated by computing a tour of lines with minimal cost. Unlike existing approaches, the tour is not forced to complete local coverage of each

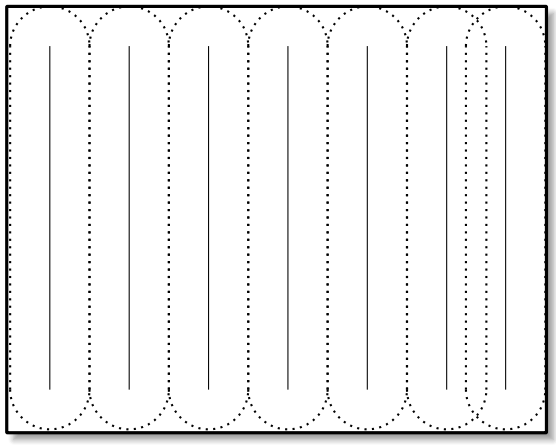

Figure 1.2: Coverage footprint over a line.

polygon before proceeding to the next.

Furthermore, this approach is used extensively in our extension to the multi-agent case. The key idea in the multi-agent case is to perform an assignment of regions to the team of n robots with the goal of minimizing the maximum cost that includes the turning costs. The process starts with any n-cell decomposition. We propose a similar method of re-evaluating cuts in this decomposition with a new cost that includes turning costs. The result is a re-optimized n-cell decomposition where each cell represents an area assigned to one of the n robots. The coverage path for each robot is computed using the minimum turns decomposition approach.

1.2 Thesis Contribution

Our key contributions for the single-agent coverage are threefold. First, we provide a method for computing the minimum altitude of a non-convex polygon, which captures the number of parallel line segments, and thus turns, needed for coverage. Second, given an initial convex decomposition of a workspace, we propose a method to iteratively re-optimize and delete cuts of the decomposition in order to optimize the altitude of the polygon on each side of the cut. The proof of its correctness and computational complexity is provided. Third, we compute a coverage path of the workspace by placing parallel line segments in each region, and then computing a minimum-length tour of the corresponding approximate convex decomposition. The tour is computed by generating and solving a Generalized Traveling Salesman Problem (GTSP) instance using existing solvers.

The key contributions for the multi-agent coverage are twofold. First, we introduce a metric that assigns an approximation of the coverage cost that includes turning costs to a polygon. Second, given an initial n-cell decomposition of a workspace, we propose an iterative procedure for re-optimizing cuts of the n-cell decomposition in order to decrease the maximum coverage cost over all cells in the decomposition. The coverage path for each robot in the team is computed using the single-agent approach.

1.3 Organization

The thesis is organized into six chapters. Chapter 2 contains the literature review of single and multi-agent coverage works. Chapter 3 reviews some of the concepts used throughout this thesis. Chapter 4 introduces our approach to the single agent coverage path planning problem. Chapter 5 introduces our approach to the multi-agent coverage path planning problem. Finally, Chapter 6 provides a conclusion to this thesis and recommendations for avenues of future research.

Chapter 2

Literature Review

The problem of coverage path planning sees many applications across numerous fields. The list includes applications like material polishing [55], autonomous farming [51], demining [3], floor cleaning [65], and many others. The roots of this problem could be traced back to CNC trajectory planning, where operators rely on their experience and intuition to plan an optimal trajectory for a tool to cut out different shapes out of a slab of metal [33]. However, the notion of coverage has been evolving since its conception. There are two common views of coverage: static and dynamic.

The static coverage objective is typically to maximize some coverage sense function. The sensor quality for these problems is a function of a distance from the robot. A common approach to these type of problem is to guide a robot to a static location in the environment that maximizes the coverage function. Examples of this include discretized workspace [24] and gradient descent methods [20].

The dynamic coverage objective is to plan paths for a robot such that all points in the workspace are covered up to a desirable level. There are a number of solutions that have been proposed for this problem. We classify them by the approach used in the solution.

One class of solutions treat this problem as a control problem. Works by Atinc [6], Hussein [41], Panagou [52] and Cassandraw [15] have developed control laws for a robot or a team of robots that achieve complete coverage of the workspace. These methods are based on gradient descent type controllers. The work by Atinc [6] fixed the issue of local minima with a improved controller.

Other classes of solutions focus on computing static paths determined by the dynamics and the workspace off-line. With these approaches, it is possible to leverage the knowledge

of the workspace and the structure of coverage paths to compute optimal paths. This thesis focuses on this approach.

Perhaps one of the earliest works in literature for static coverage paths is the work by Cao [12], where the authors outline the requirements for such paths. One of the key contributions to the field of static coverage is the work by Arkin [5]. The authors have studied two variations of the coverage problem: lawn mowing and milling. The difference is that the milling problem does not allow a robot to cross the boundary of its workspace. The authors were able to show that these problems are in fact NP-hard. Some of the other problems related to coverage are the Traveling Salesman Problem, art gallery problem, and a watchman route problem.

There are various classifications used when talking about coverage problems. Choset [18] have classified majority of the approaches by the methods used to discretized the environment. He summarized these to be either exact or approximate. However, since his survey, numerous new works have been published that do not strictly fall into these two categories. The new categories would include random and pseudo-random approaches [8, 29], 3D coverage [7, 37], coverage under uncertainty [10], persistent coverage [46], cooperative coverage [54], and others. In this section, we aim to review some of the important works involving exact and approximate decompositions as well as works that deal with the cooperative coverage.

2.1 Exact Cellular Decomposition

Typically, approaches that rely on exact cellular decomposition compute a solution in two steps. The first step involves performing some sort of decomposition of the workspace resulting in a set of cells that together make up the original workspace. The second step involves computing a path for each region separately. One of the classic examples of this is the work by Latombe [43], where a trapezoidal decomposition is performed on a polygonal planar workspace. The resulting decomposition consists of trapezoidal cells. An adjacency graph can be computed for these cells, where an edge between two vertices exists if two cells are adjacent. A tour of this graph is computed, where each cell is visited once. The coverage of each cell is by a *zig-zag* like path. In practice, this method was used on an agricultural field by Oksanen [50]. Despite the simplicity of its computation, there are numerous drawbacks to this method. One of them is the excess of cells generated, which leads to a significant overhead associated with transitioning between cells. Another problem is the direction of these zig-zag paths. This direction is determined by the decomposition, which disregards the optimality of coverage of that cell. Another problem is the generality

of this method with respect to the shape of the footprint. For example, a circular footprint would not be able to cover areas near the corners. As such, the path generated by this method would not be able to achieve complete coverage.

To mitigate the problem of excessive number of cells, Choset [18] proposed a new decomposition technique called the Boustrophedon decomposition. The Boustrophedon decomposition is very similar to the trapezoidal decomposition with the exception that some cells in the decomposition may be non-convex and the overall number of cells in the decomposition is reduced. The path planning components is identical to Latombe's approach and therefore, suffers from similar drawbacks.

Wong [64]'s landmark based decomposition is similar to the Boustrophedon but works for general workspaces. His approach is also suitable for on-line sensor based coverage. However, the individual cells are covered via the same zig-zag type paths.

Acar [2] proposed a more general decomposition technique that works on non-polygonal workspaces and could generate cells of various shapes to accommodate any coverage requirements. This decomposition is based on finding critical points of a Morse function. Boustrophedon decomposition is a specific case of a Morse decomposition, where the Morse function is a straight line. Moreover, the search for these critical points can be performed on-line leading to a number of on-line coverage algorithms [1]. The main drawback of this approach is that rectilinear workspace cannot be used.

Huang [40] studied an approach to decomposition that aims to optimize the zig-zag paths within each cell. In his work, he proposed a dynamic programming approach for choosing shapes for the cells of the decomposition in a way that minimizes the number of parallel segments of a zig-zag path within each cell. However, the transitions costs between cells are completely ignored.

Recently, in the work by Das at el. [21], a greedy decomposition of a polygonal environment was proposed that seems to result in fewer convex cells for typical workspace polygons. As with other approaches, a tour of the graph is computed. However, the inter-cell coverage of each individual cell is computed based on the graph tour. Specifically, the entry and exit points are chosen in a way that minimizes the transition costs between cells. As a result, the optimality of the intra-cell coverage within the cell is ignored. A related work by Yu [66], where the author have studied optimal traversal order for Boustrophedon type paths by solving a TSP problem.

There is a common underlying problem with exact decomposition methods. The problem of convex decomposition is known to be NP-hard for general polygons as outlined in the survey by Keil [42]. As such, most of the exact decompositions algorithms utilize heuristics. Moreover, there is a dependency between inter-cell and intra-cell coverage. An

optimal solution cannot focus on optimizing the intra-cell coverage and ignore the inter-cell transitions or vice versa.

2.2 Approximate Decomposition or Grid Based Approaches

Exact cellular decomposition is not the only available class of solutions. Another class includes grid-based methods, sometimes referred to as approximate decomposition methods. Typically, the workspace is represented as a grid of uniform cells. Such representation is easy to compute but the memory requirement scales exponentially with the size of the workspace.

Zelinsky [67] proposed a grid based coverage solution based on a wave propagation algorithm. With this method, a start and a goal cells are given. A front is created by computing the distance from the goal to the cell. Coverage is ensured by traversing the wave front, a set of cells where the values are equal. On-line generalization is available by Shrivashakar et al. [59]. A number of works [17, 28, 31, 32, 45] utilized the concepts of spanning tree coverage. The grid of the workspace acts as a graph were every cell is a node connected to its neighbors. A minimum spanning tree can be computed for slightly modified version of this graph. A coverage path is generated by tracing the edges of the minimum spanning tree, which ensures all cells are covered at least once. Arkin [5] also proposed approximate solutions based on a grid decomposition of the workspace. The workspace is represented in the form of a grid where the center point is a node in a graph. Such graph is fed into a Traveling Salesman Problem solver, where the shortest tour is computed.

All of the mentioned approaches share similar downsides. The representation of the workspace by a grid is approximate. Hence, it is expected to see some points near corners or the boundary not being covered. Furthermore, computational load for large environments makes these methods intractable. Furthermore, these methods suffer from generalization problem as well. Grid based approaches do not work well with circular coverage footprints. A robot visiting two adjacent grid cells would have to incur some overlap because the circle shape has to be larger than the grid cell. This leads to the reduction of efficiency of the coverage path and results in longer paths.

2.3 Minimizing Turns in Coverage Paths

Various works in literature study the problem of coverage with additional conditions. As an example, the work by Bretl et al. [10] studied complete coverage path generation in the presence of localization error. Others [60] looked at computing coverage paths with limited energy budget. A number of other works have looked at computing coverage paths with minimum turning. Many of the proposed solutions in literature produce paths that may not be feasible for the types of robots that perform coverage, i.e, the paths may contain turn profiles that are two aggressive. A common way to implicitly compute a path that minimizes the dynamic cost of a coverage path is by minimizing the number of turns in a coverage path.

The work by Huang [40] attempted to solve this by proposing a convex decomposition scheme via a dynamic programming approach. With this convex decomposition, the number of cells for the overall coverage is minimized. However, the time to compute such a decomposition becomes prohibitive for even medium size polygons. Arkin [4] have studied an approach, where the environment is discretized with a grid and a tour is computed with a Traveling Salesman Problem solver. However, the costs for edges connecting the grid cells are designed to incorporate turning cost. An approximate solution is proposed. Wagner [62] proposed another approximate solution with bounds. However, these two solutions restrict the type of coverage path to be rectilinear, which is overly restrictive in practice.

2.4 Multi-agent Coverage

Decreasing cost of robotics motivates the use of multi-agent systems to tackle some task. Utilizing a team of robots rather than a single robot has multiple benefits, one of which is the ability to complete a task faster. Recently, there have been a surge of works in literature for tackling robot collaboration for common tasks. In this section, we consider tasks that involve coverage.

One common approach is to divide the workspace into cells for each robot. Works by Carlsson [14] and Durham et al. [24] propose algorithms for area partitioning. The partition is computed via a sporadic pair-wise re-optimization that converges to a pair-wise optimal.

The work by Maza [47] proposed a solution to multi-agent coverage. Given n robots, the polygonal workspace is partitioned into n regions. This partition is performed via an anchored area partitioning algorithm [36]. The area of the cells in this partition reflects each robot's individual abilities. Each cell is covered in an optimal manner. The problem

is that the authors only consider convex polygons. This approach was validated with extensive trials in [9].

Hokeyam [39] worked on an algorithmic approach to coverage by robots following inner level sets of the polygonal workspace. These inner level sets are computed by shrinking the boundary of the polygon inwards by a fixed distance. A team of robots follow these level sets and resolve any collisions via sporadic communications. The idea of inner level sets is appealing for convex polygons. However, for non-convex polygons, the transitions between inner level sets could be tricky. This issue was not addressed by the authors. Moreover, this method of coverage usually ends up with excessive number of turns.

Another work by Valente [61] and Barrientos [9] propose a solution to the multi-agent problem. The solution involves several steps. The workspace is partitioned into cells with areas specified by the robots' abilities. This is performed in a distributed manner via a negotiation process. The resulting cells are discretized into a grid and a flight path is computed for each cell individually. The work by Barrientos contain extensive field tests with a team of UAVs.

Chapter 3

Background

This chapter reviews some of the concepts required to understand the proposed solution in this thesis. This chapter is organized as follows. Section 3.1 reviews robot dynamics and representation of a workspace. Section 3.2 reviews some concepts from geometry. Section 3.3 reviews graphs and the Traveling Salesman Problem.

3.1 Robot Dynamics

Throughout the thesis, we make multiple references to the coverage workspace of a robot. The coverage workspace refers to the region in the environment, such as a floor of a room, that is required to be covered. The workspace is denoted by $W \subset \mathbb{R}^2$. For the purpose of this thesis, we assume that W is connected. An example of a workspace is demonstrated in Figure 3.1.

The robot that is to be used for coverage has some dynamics. These dynamics give rise to the configuration space of a robot. The configuration space is denoted by Q. In the path planning works, an important subset of the configuration space is the free configuration space. That is, $Q_{\text{free}} \subseteq Q$ is a configuration space of a robot that is free from any collisions. Figure 3.2 demonstrates the free configuration space in orange for a robot with the shape shown in orange. It is important to note that some areas of the workspace are not in the free configuration space because the placement of a robot in such a configuration results in a collision with the boundary of the workspace. An example of such areas are the corners of the workspace. With this in mind, a feasible path is called a continuous path that is

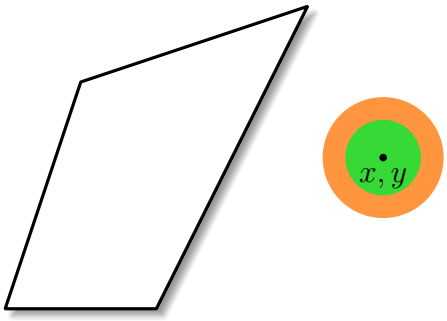

Figure 3.1: An example of a workspace with a disk robot.

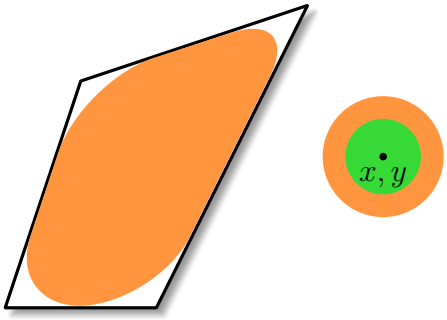

Figure 3.2: An example of a free configuration space shown in orange for a point robot.

entirely in the free configuration space:

$$p = \bigcup q_i, q_i \in Q_{\text{free}}. \tag{3.1}$$

A set of all feasible paths is denoted by P:

$$P = \{p_i \mid p_i \subseteq Q_{\text{free}}\}. \tag{3.2}$$

In this thesis, a kinematic model of Dubins' car [23] is used. The kinematic model is as follows:

$$\begin{aligned} \dot{x} &= V \cos\theta, \\ \dot{y} &= V \sin\theta, \\ \dot{\theta} &= u \end{aligned} \tag{3.3}$$

where (x, y) is a position of the robot and V is the linear constant speed of a car, u is the turn rate, which is bounded. The maximum turning rate corresponds to the minimum turning radius R_{min}. With this model, the configuration space of a car is $R^2 \times \Theta$

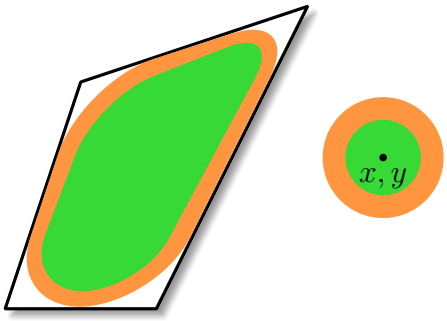

Figure 3.3: An example of a coverable space show in green.

Throughout the thesis, numerous references are made to straight line segments and transition segments. A straight line segment is a path that is a straight line. In other words, a robot traversing a straight line segments will never change its heading. On the other hand, a transition segment is a path where the robot does change its heading. A path p has a linear component $\ell(p)$ and an angular component $a(p)$. The linear component is the total distance traveled by a robot traversing p. The angular component is an accumulation of heading changes of a robot traversing p.

Any coverage robot is equipped with a coverage tool. When this tool is operational, there is an associated coverage footprint with it. The location of this footprint and the exact region covered by this footprint depend on the physical location of this footprint on the robot. To formalize this relation, a function is introduced called the coverage map, $\mathcal{M} : Q \to \mathbb{R}^2$. The coverage map function maps the configuration q of a robot to a set of points in $\mathbb{R}^2$ that are located under the footprint. A set of points that can be covered given a free configuration space of a robot is called the coverable space and is denoted by $\mathbb{C}$. An example of a coverable space is shown in Figure 3.3. In the figure, a disk robot has a smaller circular coverage footprint in the center of the robot shown in green. Formally, the coverage space is defined as follows:

$$\mathbb{C} = \bigcup_{q \in Q_{\text{free}}} \mathcal{M}(q). \tag{3.4}$$

3.2 Geometric Preliminaries

A *simple polygon* is a non-intersecting chain of straight line segments forming a closed loop and specified as a set of points, i.e., $Z = \{v_i \in \mathbb{R}^2 | i = 1, \ldots, n\}$. Note that since

the chain is a circuit, any $v_i \in Z$ has two adjacent line segments. A *boundary of a simple polygon* is a set of points, ∂Z, along a line connecting any two adjacent vertices of a chain. A *clockwise simple polygon* is a simple polygon where vertices are specified in a clockwise order. We associate these types of simple polygons with holes in the workspace. A *counter-clockwise simple polygon* is a simple polygon where vertices are specified in a counter-clockwise order. This type of simple polygons are associated with the external boundary of the workspace. Finally, a *polygon* is a set of clockwise and counter-clockwise simple polygons. For clarity, we will refer to simple polygons as chains and reserve the term polygon for a set of chains, i.e., $P = \{Z_0, \ldots, Z_m\}$. Z_0 is a counter-clockwise chain (i.e., the boundary) and all subsequent Z_i are clockwise chains (i.e., holes). A *boundary of a polygon* is the set of points defined as $\partial P = \bigcup_{i=1}^{M} \partial Z_i$ where $Z_i \in P$.

A *reflex vertex* is a vertex in one of the chains of P that has an internal angle of more than π. As such, a polygon is *convex* if and only if there are no reflex vertices. Conversely, a polygon is non-convex if and only if it contains at least one reflex vertex. Note that the presence of a hole guarantees at least one reflex vertex.

Definition 3.2.1 (Cone of bisection)**.** Suppose a non-convex polygon P containing a reflex vertex v is given. Let $d_\ell = \{v_k, v\}$ and $d_r = \{v, v_l\}$ be the two adjacent edges of v. The *cone of bisection* at v is defined by two line segments, $e_\ell = \{v, w_\ell\}$ and $e_r = \{v, w_r\}$ that are parallel to d_r and d_ℓ respectively with $w_\ell, w_r \in \partial P \setminus \{v\}$. See Figure 3.4.

Definition 3.2.2 (Cut space)**.** Consider a non-convex polygon P with reflex vertex v, along with the cone of bisection. The cut space is a set $S \subset \partial P$ of all points on ∂P between w_ℓ and w_r visible from v.

The cut space S can be represented by its straight line segments $S_1, \ldots, S_L$ where $\bigcup_{i=1}^{L} S_i \subseteq S$.

Definition 3.2.3 (Decomposing Cut)**.** Given a non-convex polygon P with a reflex vertex v, a decomposing cut is a straight line segment $e = \{v, w\}$ within the cone of bisection of v where v, w belong to the same chain. See Figure 3.4.

3.3 Traveling Salesman Problem Preliminaries

A graph G is defined by a pair of sets $G = (V, E)$, where V is a set of vertices in the graph and E is a set of edges between the vectices. An edge connecting vertices u and v exists in the graph if there is a pair $\{u, v\}$ in E. A graph is complete if there exist a pair $\{u, v\}$ for every $u, v \in V, u \neq v$. An example of a complete graph is shown in Figure 3.5.

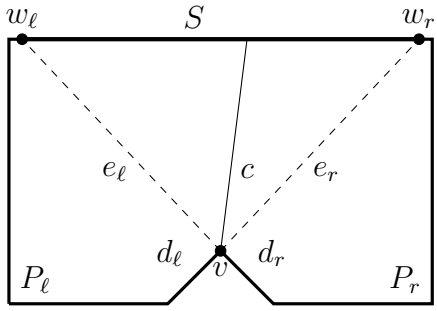

Figure 3.4: Decomposing cut originating at a reflex vertex.

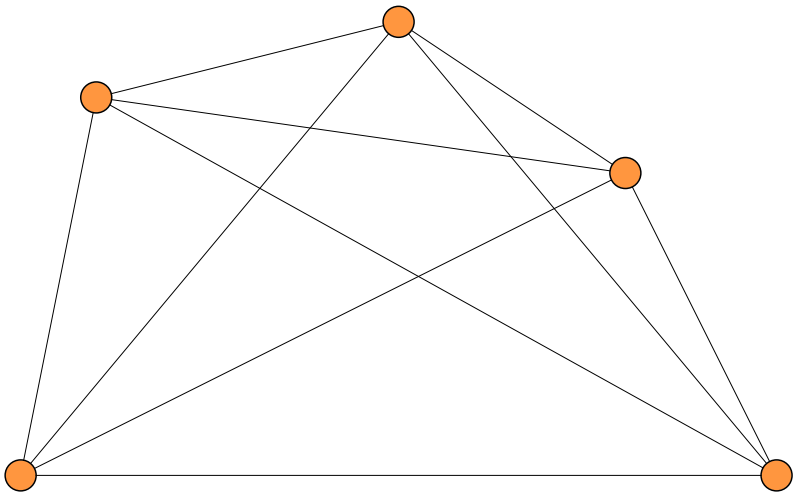

Figure 3.5: An example of a complete graph G.

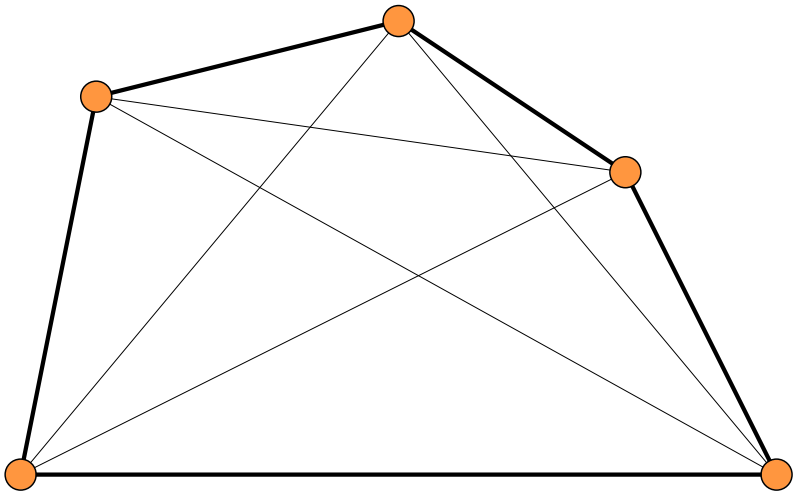

Figure 3.6: An example of a TSP tour on a complete graph G.

Definition 3.3.1 (Weighted Graph)**.** A graph G is weighted if for every edge $e_i \in E$, there exists a value $w_i > 0$.

Definition 3.3.2 (Subgraph)**.** A subgraph $\mathcal{G} = (V_S, E_S)$ of a graph $G = (V, E)$ is a graph where $V_S \subseteq V$ and $E_S \subseteq E$. The subgraph is a spanning subgraph when $V_G = V$.

Definition 3.3.3 (Cycle)**.** A cycle C in graph G is a sequence of vertices $v_1, v_2, \ldots, v_k$, such that there is no repeating vertices in the sequence and no repeating edges. The last edge of a cycle is an edge between v_1 and v_k.

Definition 3.3.4 (Tour)**.** A cycle on a spanning subgraph is called a tour.

Problem 3.3.1 (TSP)**.** *Given a complete weighted graph $G = (V, E, w)$, compute a tour T of G such that the sum of edge weights is minimized. An example of a TSP tour is shown in Figure 3.6.*

Another important problem is called the Generalized Traveling Salesman Problem (GTSP). It is defined over a complete graph and a vertex partition. A vertex partition is a set of disjoint subsets of V, $\{V_1, V_2, \ldots, V_m\}$.

Problem 3.3.2 (GTSP)**.** *Given a complete graph $G = (V, E, w)$ and a vertex partition $\{V_1, V_2, \ldots, V_m\}$, compute a tour of G that includes exactly one vertex from each subset V_i and the sum of edge weights is minimized. An example of a GTSP tour is shown in Figure 3.7.*

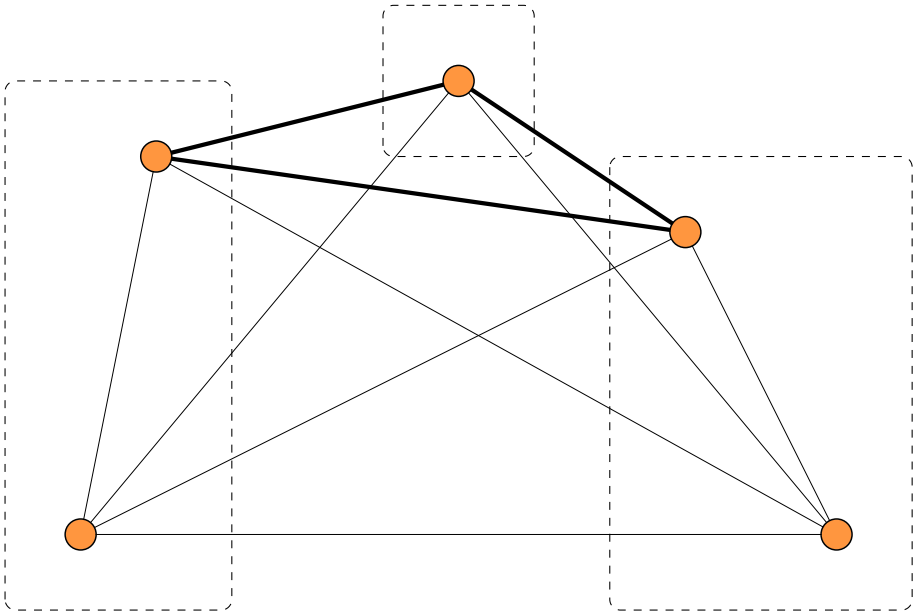

Figure 3.7: An example of a GTSP tour on a complete graph G.

A TSP problem is known to be NP-hard. A GTSP problem can be solved as a reduction to a TSP problem [48]. However, there are numerous approximate and suboptimal solutions available. One of them is a solver called GLKH, which utilizes a Lin-Kernighan TSP heuristic [35]. In this thesis, we utilize the GLKH solver for solving GTSP problems.

Chapter 4

Single Agent Coverage

In this chapter, the problem of single agent coverage is formulated and a solution is proposed. The chapter is organized into six sections. Section 4.1 goes through a formulation of the problem. Section 4.2 proposes an algorithm. Section 4.3 performs a computational complexity analysis. Section 4.4 describes the formulation of the problem as a Generalized Traveling Salesman Problem. Lastly, Section 4.5 presents the results of the algorithm in the form of simulations.

4.1 Problem Statement

The general coverage problem is defined in Problem 4.1.1.

Problem 4.1.1 (Minimum Cost Coverage Path Planning)**.** *Given a workspace W, a robot with dynamics, and a coverage footprint map $\mathcal{M}(q)$, compute a path p_i such that*

$$\begin{aligned} & p_i \in P, \\ & \bigcup_{q \in p_i} \mathcal{M}(q) = \mathbb{C}, \\ & \mathcal{E}(p_i) \leq \mathcal{E}(p_j), \forall p_j \in P. \end{aligned} \tag{4.1}$$

In Problem 4.1.1, a workspace W, a robot with dynamics, and a coverage footprint map are given. The objective is to compute a path that lies entirely in the free configuration space of the robot such that the footprint of the whole path is equal to the coverable

space and the cost of the path is minimized. A number of aspects of this problem makes it difficult. For instance, the set of feasible paths W contains infinitely many paths. Also, it is difficult to directly check the satisfiability of conditions (4.1). As such, several assumptions and heuristics are used throughout this work.

Assumption 4.1.1 (Polygonal Workspace)**.** *The workspace is assumed to be polygonal,simply connected, and noise free.*

Assumption 4.1.2 (General Robot Model Assumption)**.** *We deal with kinematic robot models with $v_{\max}$ as the maximum velocity. In Section 4.5, we utilize Dubins' car model for our simulations. We assume robot's coverage footprint is a circle of radius r. The localization problem is assumed to be solved.*

4.1.1 Utilizing Segmented Paths

Assumption 4.1.3 (Reduced Set of Feasible Paths)**.** *It is assumed that paths considered for coverage in this problem have a certain structure. The set of such paths is called a set of feasible segmented paths.*

The set of feasible segmented paths is defined in Definition 4.1.1 with an example shown in Figure 4.1. Notably, these types of paths consists of a sequence of pairs of straight and transition segments.

Definition 4.1.1 (Set of Feasible Segmented Paths)**.** A set of feasible segmented paths is defined as

$$P_{\text{segmented}} = \{p_i \mid p_i \subseteq Q_{\text{free}},\ p_i = \{s_1, t_1, s_2, t_2, \ldots, s_n, t_n\}\}, \tag{4.2}$$

where s_i refers to i^{th} straight line segment and t_i refers to i^{th} transition segment.

Problem 4.1.1 is modified accordingly with a modified search space of feasible paths.

Problem 4.1.2 (Minimum Cost Coverage Path Planning with Straight Line Segments)**.** *Given a workspace W, a robot with dynamics, and a coverage footprint map $\mathcal{M}(q)$, compute a path p_i such that*

$$\begin{aligned} & p_i \in P_{segmented}, \\ & \bigcup_{q \in p_i} \mathcal{M}(q) = \mathbb{C}, \\ & \mathcal{E}(p_i) \leq \mathcal{E}(p_j), \forall p_j \in P_{segmented}. \end{aligned} \tag{4.3}$$

4.1.2 Path Evaluation Metric

From the literature and our collaborations with companies that deal with coverage, we have observed that the most desirable path for numerous types of robots is a straight line segment. Human path planners generally try to avoid having excessive number of turns in their paths. This is because many types of robotic systems do not handle turns well. For instance, a battery powered differential drive robot often cannot maintain the same linear speed when executing turn maneuvers. As a result, it needs to slow down, turn and speed up afterwards, which penalizes the energy efficiency of the path. Other examples include field scanning via a fixed wing UAV [27]. Such fixed wing UAV systems often have the sensor suite mounted directly on to the chassis of the plane for cost and weight considerations. When the plane is not pitching, the sensors point downwards. During a turn maneuver, the UAV has to pitch, which cause the sensors to point in an other direction. This reduces the quality of the coverage and usually requires a second pass over the uncovered spot. Again, turns reduce the efficiency of coverage paths. Motivated by these observations, we aim to compute coverage paths with minimum turning.

Note that any path p_i has two components: linear and angular. A metric to evaluate the cost of a path is designed to reflect the differences in the cost of a straight path versus a curved path. The cost function for path p_i is denoted by $\mathcal{E}(p_i)$ and defined as follows:

$$\mathcal{E}(p_i) = c_1\ell(p_i) + c_2a(p_i). \tag{4.4}$$

Here, $\ell(p_i)$ is the length of a path, $a(p_i)$ is the total cumulative angle of the path, c_1 and c_2 are constants.

Assumption 4.1.4 (Turn Penalizing Dynamics)**.** *It is assumed that the robot has particular dynamics that heavily penalizes turning costs resulting in*

$$c_2 \gg c_1. \tag{4.5}$$

In other words, for such a robot, the best path between any two points is a straight line segment if it is feasible.

Remark 4.1.1 (Angular Component of Straight vs. Transition Segments). Observe that the cost of a straight line segment s_i has only the linear component

$$\mathcal{E}(s_i) = c_1\ell(s_i). \tag{4.6}$$

Whereas the cost of a transition segment t_i typically has both linear and angular components

$$\mathcal{E}(t_i) = c_1\ell(t_i) + c_2a(t_i). \tag{4.7}$$

•

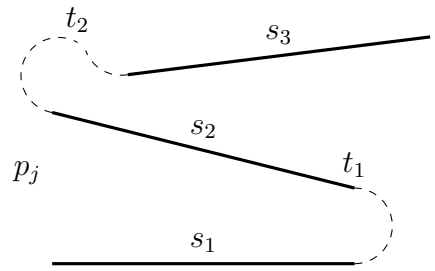

Figure 4.1: An example of a segmented path.

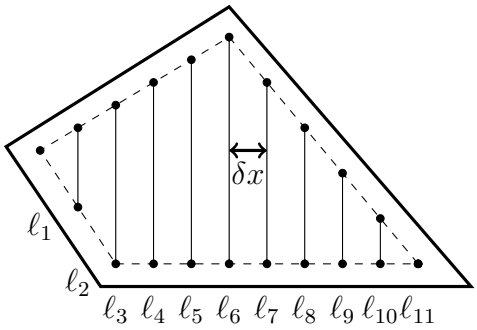

Figure 4.2: Parallel line arrangement.

Note that the cost for a segmented path has the following structure:

$$\mathcal{E}(p_i) = \sum_{i=1}^{n} (\mathcal{E}(s_i) + \mathcal{E}(t_i)) = \sum_{i=1}^{n} [c_1 \ell(s_i) + c_1 \ell(t_i) + c_2 a(t_i)]. \tag{4.8}$$

4.1.3 Subclass of Segmented Paths: Boustrophedon

Suppose a coverage robot is traversing a straight line segment of length ℓ_i. Suppose the width of robot's coverage footprint is δx. The area covered after the robot finishes traversing the line is $\ell_i \delta x + k$ where k is the amount of area covered beyond the starting and end points of the line. Suppose there are n such lines with no overlap between their line footprints stacked parallel to each other. For this exercise, assume the robot is able to teleport from one line to another. Such an arrangement is commonly used in the coverage literature and is commonly referred to as Boustrophedon paths [19]. An example is shown in Figure 4.2.

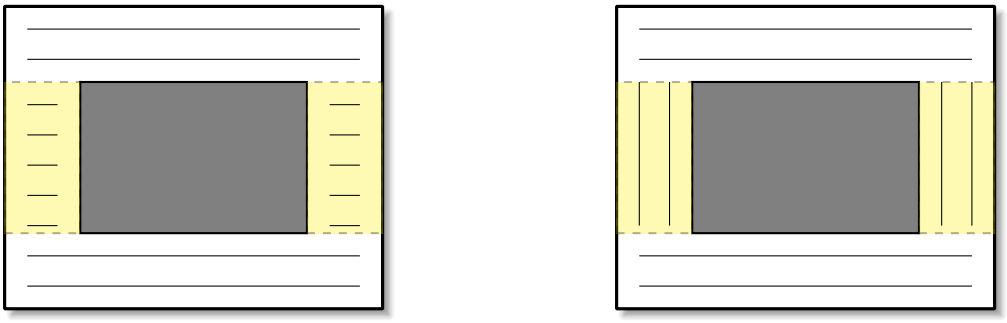

Figure 4.3: Polygon regions where lines can be reoriented.

Such an arrangement has some desirable properties such as having no overlap between straight line segments, having most of the interior of the workspace covered by straight line segments, and transitions are performed near the borders of the workspace. The total area covered by this path can also be computed as

$$A_{\text{covered}} = \sum_{i=1}^{n} \ell_i \delta x + k. \tag{4.9}$$

Furthermore, this arrangement holds some similarities to Riemann sums. As such, as δx gets smaller, which happens when a small robot is covering a huge workspace, A_{covered} tends to the actual area of the workspace, which is a desirable property. And lastly, all transition segments in such an arrangement have a nearly fixed cost. This is because every transition segment is a transition from the end of one line to the start of the next line. Such a transition has an 180^o angular component associated with it.

However, several issues prevent us from using this arraignment is a sole solution to the coverage problem. The first issue is related to the coverage near the borders of the workspace. Some areas cannot be covered by this path alone and usually requires a wall following path. Second issue is with choosing the orientation of these parallel line segments. Some orientations are more optimal than others. And lastly, for concave workspaces, this type of coverage is non-optimal. See Figure 4.3 for an example. Nevertheless, the solution proposed in this work utilizes some aspects of Boustropheon paths.

4.1.4 Coverage with Minimum Turns

Suppose a Boustropheon path is to be used to cover a convex polygon. Recall the structure of the cost function of a segmented path from equation(4.8) and the property of Boustrophedon paths that state that the transition segments have a fixed cost. Since the number

of straight line segments is the same as the number of transition segments, the problem becomes one of minimizing the number of straight line segments such that complete coverage is achieved.

To this end, a notion of altitude is introduced. This notion of the altitude first appeared in the work by Huang [40]. For convex polygons, the altitude in direction θ is defined as the minimum distance between two parallel lines angled at $\theta + \frac{\pi}{2}$ with respect to the x-axis such that all vertices of the convex polygon are contained between these two lines [40]. This notion can be extended to general polygons. The method of obtaining θ altitude for a general polygon is shown in Algorithm 1. Figure 4.4 provides an example of this procedure. In the example, the altitude is $x_1 + 2x_2 + 3x_3 + 2x_4 + x_5$. Note that if r is the radius of the footprint of the robot and α is the altitude of a polygon along θ then the total number of lines required for complete coverage in direction θ is

$$n = \left\lceil \frac{\alpha}{2r} \right\rceil. \tag{4.10}$$

Moreover, because of the structure of the Boustrophedon paths, the altitude also provides the angular component of the whole path. Specifically,

$$a(p_i) = 180^o \left\lceil \frac{\alpha}{2r} \right\rceil. \tag{4.11}$$

Given a polygon W, we are interested in finding the minimum altitude $\alpha^*(W)$. Note, however, that the altitude can be measured with respect to an infinite number of directions θ. The following result addresses this issue.

Proposition 4.1.1 (Minimum Altitude Directions, [40])**.** *Given a polygon W, the direction of minimum altitude is orthogonal to one of the edges of the polygon.*

By Proposition 4.1.1, minimum altitude can be computed by performing Algorithm 1 on each direction associated with the edge of the polygon. The runtime of this approach is $O(n^2 \log n)$ in number of vertices of a polygon.

4.1.5 Decoupled Problem Statement

As stated before, the Boustrophedon paths are not optimal for non-convex polygons. However, it can act as a starting point in an optimization. A more sophisticated approach is to decompose W into polygons $w_0, w_1, \ldots, w_k$, where for each w_i, the coverage is achieved

Algorithm 1 get_general_altitude(W, θ)

Require: Polygon $W = \{Z_0, Z_1, \ldots\}$, direction θ.

counter $\leftarrow 0$, $\alpha \leftarrow 0$
Rotate W by $-\theta$ to align direction with x-axis.
Sort all vertices in W by their x-coordinates
for each v_i in the sorted list **do**
  $\alpha \leftarrow \alpha +$ counter $\times (x_{v_i} - x_{v_{i-1}})$
  if both vertices adjacent to v_i are on right **then**
    Increment counter by 1
  else if both vertices adjacent to v_i are on left **then**
    Decrement counter by 1
  end if
end for
return α

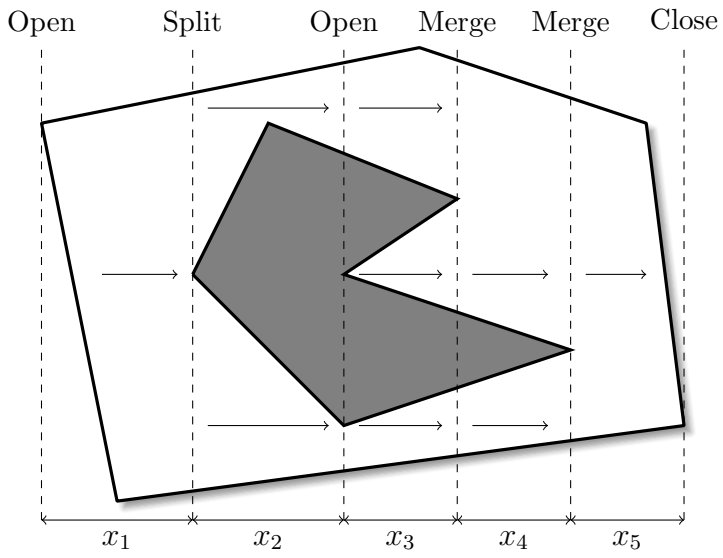

Figure 4.4: Process of measuring altitude with θ equal to 0.

through the Boustrophedon path. The straight line segments in each such path should be oriented to minimize the number of lines. For example, the polygon in Figure 4.3 is decomposed into four regions. Lines in the two regions on the sides of the polygon are reoriented to reduce the number of lines in those regions.

With the notion of altitude for general polygons, the problem statement becomes as follows.

Problem 4.1.3 (Min Altitude Decomposition)**.** *Given a polygon W, find k and $w_1, w_2, \ldots, w_k$ such that $\sum_{i=1}^{k} \alpha^*(w_i)$ is minimized where $\bigcup_{i=1}^{k} w_i = W$.*

For each w_i, a number of parallel lines is generated in an orientation that minimizes the altitude. The result is a set of lines, Γ. Now the problem becomes that of choosing an order and direction to traverse each line with, i.e., a tour of Γ. Observe that each line can be traversed with two directions. Hence, the second subproblem can be defined as follows:

Problem 4.1.4 (Minimum Cost Tour)**.** *Given a polygon W and a set of lines Γ, generate a matrix M of transition costs between any pair of elements of Γ. Find a tour of Γ such that cost of the tour is minimized.*

The following sections will introduce our approach to these problems.

4.2 Minimum Altitude Polygon Decomposition for Coverage Planning

This section introduces our algorithm to Problem 4.1.3.

4.2.1 Minimum Altitude Decomposition

This section outlines the approach to Problem 4.1.3. The overall steps of the process are described in Algorithm 2.

Algorithm 2 accepts a polygon as the input. On Line 1, initial decomposition is generated using any convex decomposition technique. The rest of the algorithm attempts to re-optimize this decomposition in order to decrease the overall altitude. Re-optimization steps on Lines 3 and 4 are performed until the cost of the decomposition no longer decreases. The core of Algorithm 2 is the procedure for making optimal decomposing cuts shown in Line 3. The procedure for making optimal decomposing cuts is introduced in Section 4.2.2.

Algorithm 2 min_alt_decomposition(W)

Require: Polygon $W = \{Z_0, Z_1, \ldots\}$
1: $D \leftarrow$ any convex decomposition of W
2: **repeat**
3: Re-optimize a cut from D
4: Update cost of the decomposition
5: **until** stopping criteria is met

4.2.2 Optimal Decomposing Cut

The re-optimization step operates on a polygon with a reflex vertex. This polygon is formed by removing previous cuts that were made by a convex decomposition. Our algorithm will either generate a new cut or no cut depending on what is optimal with respect to the altitude of the polygon. We do this by searching for optimal decomposing cuts. The algorithm for doing that is shown in Algorithm 3. The procedure operates on a single chain and a specified reflex vertex in that chain. On Line 2, a cut space is generated for the reflex vertex, which provides a set of potential cuts. Two polygons, W_ℓ and W_r, are formed by initializing the cut at the first point of the cut space on Line 3. Two sets of altitude directions are generated on Lines 4-5 based on the two polygons. The main loop on Lines 6-17 locates candidates for a cut for each combination of directions and each straight segment of the cut space. Lines 9-10 locate special points on S_i called transition points for each altitude direction. These points are points on S_i that yield cuts minimizing the sum of altitudes. Transition points are found by locating a point of intersection of S_i and a line h_i parallel to the edge e_i passing through v (left of Figure 4.6). Before moving on to the next straight segment of the cut space, W_l is modified to include S_i in its boundary. W_r is modified to exclude S_i from its boundary. The straight segment S_i may also add a new altitude direction θ, which has to be accounted for in Θ_ℓ and Θ_r. These operations are carried out on Lines 15-17. The algorithm returns the cut that has the lowest cost.

4.2.3 Proof of Correctness

Algorithm 3 provides an optimal decomposing cut. This section proves this claim.

Claim 4.2.1. *Given a polygon W and a reflex vertex v, Algorithm 3 returns a decomposing cut forming two new polygons that minimize $\alpha^*(W_\ell) + \alpha^*(W_r)$.*

The rest of this section will provide proof for this claim. Suppose a polygon W is given

Algorithm 3 find_optimal_cut(W, v)

Require: Polygon $W = \{Z_0\}$, reflex vertex v

$A_\ell \leftarrow \emptyset,\ A_r \leftarrow \emptyset,\ U \leftarrow \emptyset$
Find cut space S for v
$W_\ell, W_r \leftarrow$ polygons after the cut at first endpoint of S
$\Theta_\ell \leftarrow$set of directions θ orthogonal to edges of W_ℓ
$\Theta_r \leftarrow$set of directions θ orthogonal to edges of W_r
for each $S_i \in S$ **do**
  for each $\text{dir}_\ell \in \Theta_\ell$ **do**
    for each $\text{dir}_r \in \Theta_r$ **do**
      $d_\ell \leftarrow$ transition point on S_i w.r.t dir_ℓ
      $d_r \leftarrow$ transition point on S_i w.r.t dir_r
      Add tuple $(d_\ell, \text{dir}_\ell, \text{dir}_r)$ to U
      Add tuple $(d_r, \text{dir}_\ell, \text{dir}_r)$ to U
    end for
  end for
  Modify W_ℓ by adding S_i to its boundary
  Modify W_r by removing S_i from its boundary
  Modify Θ_ℓ, Θ_r if new θ was introduced by S_i
end for
Compute costs for all tuples $(u, \text{dir}_\ell, \text{dir}_r) \in U$
return Lowest cost element from U

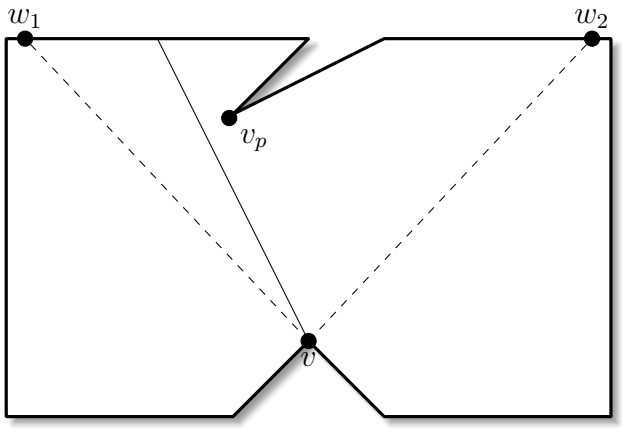

Figure 4.5: An example of a pinch vertex at v_p.

with a reflex vertex v. Recall that a decomposing cut generates two new polygons, W_ℓ and W_r. The sets of all edges of W_ℓ and W_r are E_ℓ and E_r respectively. We are interested in studying how a choice of a cut affects the altitudes of W_ℓ and W_r. Let us parameterize the orientation of a cut with respect to t. There is a function $f : \mathbb{R} \to \mathbb{R}^2$ that maps a parameter $t \in [0, 1]$ to a point $x \in \mathbb{R}^2$ on the cut space. Let w_1 be a point on cut space when $t = 0$ corresponding to a cut on edge e_1. See Figure 4.5. Define $\alpha_i(t)$ as a function that measures altitude orthogonal to edge e_i.

Define the two sets of altitudes as follows:

$$\begin{aligned} A_\ell &= \{\alpha_{\ell,i}(t) \mid i \in \{1, \ldots, |E_\ell|\}, \\ A_r &= \{\alpha_{r,i}(t) \mid i \in \{1, \ldots, |E_r|\}. \end{aligned} \tag{4.12}$$

Note, each element $\alpha_{\ell,i}(t) \in A_\ell$ and $\alpha_{r,i}(t) \in A_r$ is a semi-continuous function of t, with discontinuities occurring only when the cut sweeps past a pinch vertex in the cut space as shown in Figure 4.5.

Lemma 4.2.1. *There exists a $t \in [0, 1]$ that minimizes $\alpha_{\ell,i}(t) + \alpha_{r,j}(t)$.*

Proof. For W_ℓ and W_r, the minimum altitudes are

$$\begin{aligned} \alpha_{1,\min}(t) &= \min(\alpha_{1,1}(t), \ldots, \alpha_{1,n}(t)), \\ \alpha_{2,\min}(t) &= \min(\alpha_{2,1}(t), \ldots, \alpha_{2,m}(t)). \end{aligned} \tag{4.13}$$

These functions are semi-continuous. Furthermore, the objective of the optimization is the sum of two minimum altitudes of the two polygons,

$$\alpha_{1,\min}(t) + \alpha_{2,\min}(t). \tag{4.14}$$

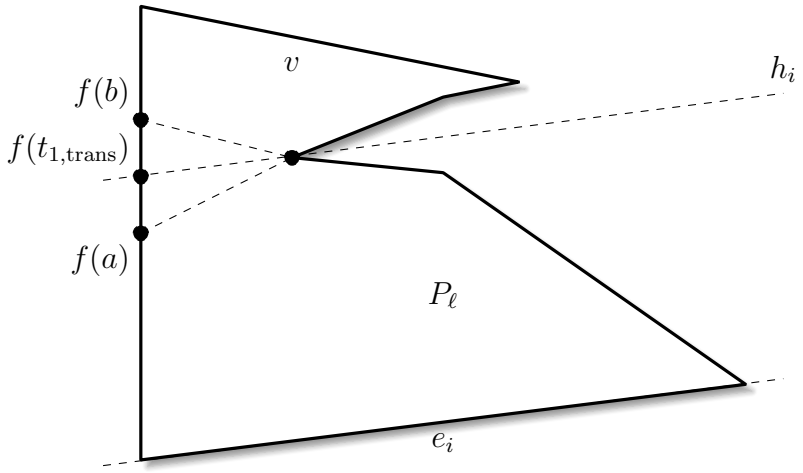

Figure 4.6: An example of a transition point.

The function in Equation 4.14 is also semi-continuous. Furthermore, since $t \in [0, 1]$ is in a compact set, the minimum of Equation 4.14 exists by the extension of the extreme value theorem. □

Fix $S_i \in S$ and let $f(a)$ and $f(b)$ correspond to end points of S_i as shown in Figure 4.6. Consider $\alpha_{1,i}(t) \in A_\ell$, which is measured with respect to e_i. Construct a line h_i that is parallel to e_i that passes through v. Denote $f(t_{1,\text{trans}})$ as the point on S_i that intersects h_i. Let us refer to this point as a transition point. There are three possibilities:

1. $t_{1,\text{trans}} \geq b$,
2. $t_{1,\text{trans}} \leq a$,
3. $t_{1,\text{trans}} \in (a, b)$.

These cases are shown in Figure 4.7. In the first case, $\alpha_{1,i}(t)$ remains constant for all $t \in [a, b]$. In the second case, $\alpha_{1,i}(t)$ is an increasing linear function for all $t \in [a, b]$. In the third case, $\alpha_{1,i}(t)$ remains constant for all $t \in [a, t_{1,\text{trans}}]$ and becomes an increasing linear function for all $t \in [t_{1,\text{trans}}, b]$.

Now, we perform similar analysis for the same S_i but for the right polygon W_r. The results are almost identical where the only difference is the behavior of $\alpha_{2,i}(t)$ in three cases outlined above. Figure 4.8 demonstrates these cases. In the first case, $\alpha_{2,i}(t)$ is a decreasing linear function for all $t \in [a, b]$. In the second case, $\alpha_{2,i}(t)$ remains constant for all $t \in [a, b]$.

In the third case, $\alpha_{2,i}(t)$ is a decreasing linear function for all $t \in [a, t_{2,\text{trans}}]$ and remains constant for all $t \in [t_{2,\text{trans}}, b]$. Hence, for each S_i and a combination of altitudes, $\alpha_{1,i}(t)$ and $\alpha_{2,j}(t)$, we have two transition points, $\{t_{1,\text{trans}}, t_{2,\text{trans}}\}$.

Lemma 4.2.2. *For a given S_i, altitudes $\alpha_{1,i}(t)$ and $\alpha_{2,j}(t)$, a minimizer of $\alpha_{1,i}(t) + \alpha_{2,j}(t)$ is $t \in \{t_{1,\text{trans}}, t_{2,\text{trans}}\}$.*

Proof. Observe that $\alpha_{1,i}(t) + \alpha_{2,j}(t)$ is a sum of piece-wise linear functions. For any combination of two cases depicted in Figure 4.7 and Figure 4.8, there are two cases to consider when looking for the minimizer of the sum.

If $t_{1,\text{trans}} \geq t_{2,\text{trans}}$ then the optimal happens at $t_{2,\text{trans}}$. If $t_{1,\text{trans}} < t_{2,\text{trans}}$ then the optimal depends on the slopes of the linear functions. Let $\delta\alpha_\ell$ and $\delta\alpha_r$ be the slopes of the left and right altitudes respectively. If $\delta\alpha_\ell > \delta\alpha_r$ then the optimum point is at $t_{1,\text{trans}}$. Otherwise, it is at $t_{2,\text{trans}}$.

There is an altitude associated with the cut itself in A_ℓ and in A_r that does not exhibit the same properties that were discussed previously. However, by the properties shown by Huang [40], this altitude varies as a piece-wise continuous sinusoidal function of ϕ where ϕ is the angle of a cut with respect to e_1. This means the altitude for W_ℓ is minimized when either the cut is parallel to one of the edges of W_ℓ or when $t \in \{a, b\}$. Likewise for W_r. However, when the cut is parallel to one of the edges of W_ℓ or W_r, this means that the altitude is identical to another altitude in the set A_ℓ or A_r. Since these altitudes are checked first, we only have to check both transition points at $f(a)$ and $f(b)$. □

By Lemma 4.2.2, it is enough to compute the cost of each transition point for each $S_i \in S$ and all combinations of elements from A_ℓ and A_r and pick one point with lowest cost. This is the procedure described in Algorithm 3.

4.3 Computational Complexity

The core of the algorithm is Algorithm 3, where an optimal cut is computed. We start the computational analysis with this algorithm. On Line 2, a cut space is computed. Its computation is done by computing a visibility polygon centered around the reflex vertex. An efficient $\mathbb{O}(n)$ algorithm for computing a visibility polygon is proposed in [25] where n is the number of vertices in a polygon. Our implementation utilizes the Visilibity [49] library. Line 3 performs a cut of a polygon resulting in two new polygons. This step is performed in $\mathbb{O}(n)$ since it only involves a walk through the chain of a polygon. On Lines 4

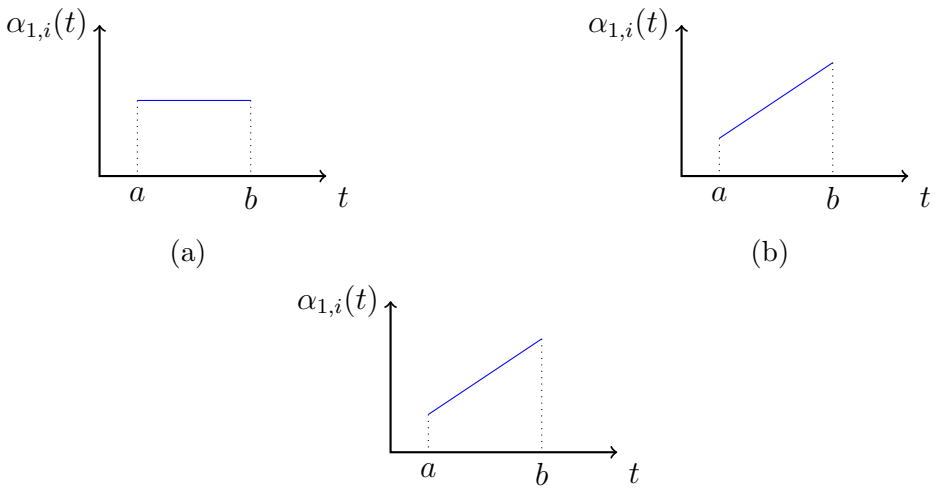

Figure 4.7: Three cases of altitude behavior for W_ℓ.

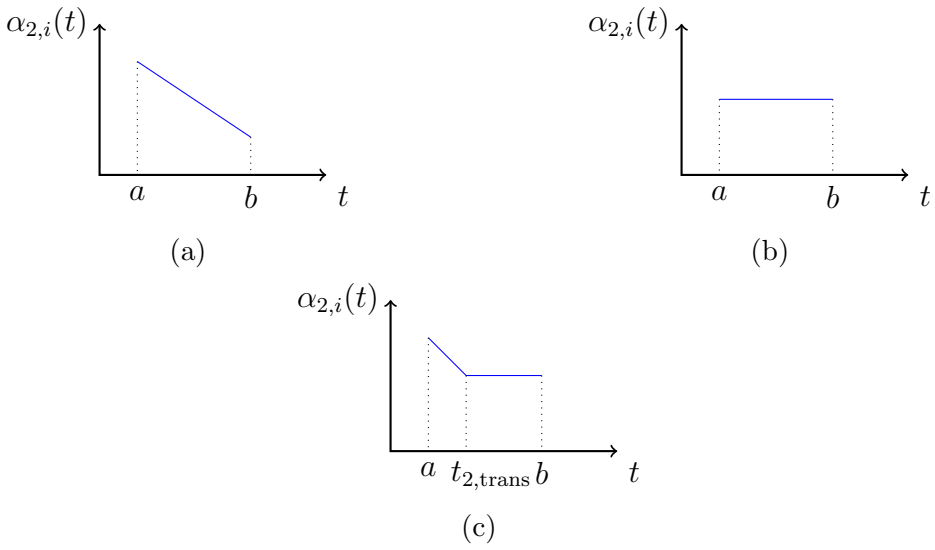

Figure 4.8: Three cases of altitude behavior for W_r.

- 5, two sets of angles is generated. These steps are both $\mathbb{O}(n)$ in the worst case since all edges of both polygons need to be checked. On Line 9 - 10, transition points are computed. Computing an intersection point of two lines is $\mathbb{O}(1)$. Since we are checking all edges on the boundary, it is $\mathbb{O}(n)$. The inner-most loop iterates n times in the worst case. Same for the middle loop. Hence, the two loops run in $\mathbb{O}(n^3)$. The cardinality of U after the two inner loops is $\mathbb{O}(n^3)$. Lines 15- 17 run in $\mathbb{O}(1)$ since the endpoints of S_i are indexed and inserting them into the chain is done in constant time. Hence, the outer-most loop runs in $\mathbb{O}(n^4)$. Finally, Line 19 finds the element with the lowest cost. This means that the altitude has to be computed for each element. The altitude as shown in Algorithm 1 has a run time of $\mathbb{O}(n^2 \log n)$. As an input, the altitude algorithm accepts a polygon. This means a polygon has to be constructed for each element in U, which takes $\mathbb{O}(n)$. Therefore, for each element in U, the run time for all computations is $\mathbb{O}(n^3 \log n)$. Since there are $\mathbb{O}(n^3)$ elements in U, this step of the algorithm runs in $\mathbb{O}(n^6 \log n)$ time. The overall run time for Algorithm 3 is $\mathbb{O}(n^6 \log n)$.

The optimization procedure shown in Algorithm 2 has a stopping criteria, which is determined by the polygon, the quality of the initial decomposition, and the size of the polygon. As such, the run time cannot be stated. However, since the acceptance conditions for any new cut is that it is strictly better than the previous cut, the convergence of Algorithm 2 is guaranteed.

4.4 GTSP Tour Generation

Given a polygon W, Algorithm 2 is called on W, resulting in a set D of polygons. Each $W_i \in D$, is populated with a minimum set Γ_e of straight line segments. Note, each line $\gamma_j \in \Gamma_e$ has two possible traversal directions. We refer to the choice of this direction as γ_j^1 or γ_j^2 as shown in Figure 4.9.

Define $\text{cost}(\gamma_i^m, \gamma_j^l)$ as the robot transition cost from line γ_i in m direction to region γ_j in l direction, where $m, l \in \{1, 2\}$ and $i, j \in \{1, \ldots, k\}, i \neq j$. This transition cost takes the dynamics of the robot into account. By computing this cost for each pair γ_i^m, γ_j^l, a complete graph $G = (V, E, w)$ is constructed.

In this graph, a vertex $v \in V$ represents one possible choice of direction to traverse a line, γ_i^m for some $m \in \{1, 2\}$ and $i = \{1, \ldots, k\}$. An edge $e = \{v, z\}$ in E represents transition between vertices v and z, and weights represent robot transition costs between vertices. Furthermore, V is partitioned into sets $\{\gamma_i^1, \gamma_i^2\}$ for $i = \{1, \ldots, k\}$. The graph G and the partition define the GTSP instance, and a GTSP tour on this graph gives a coverage path for the robot. This process is demonstrated in Figure 4.10.

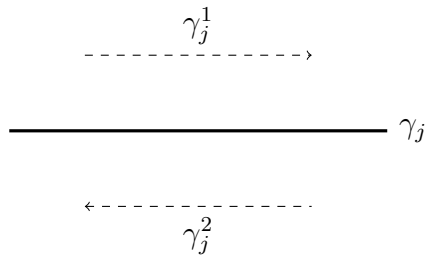

Figure 4.9: Two possible traversal directions for line γ_j.

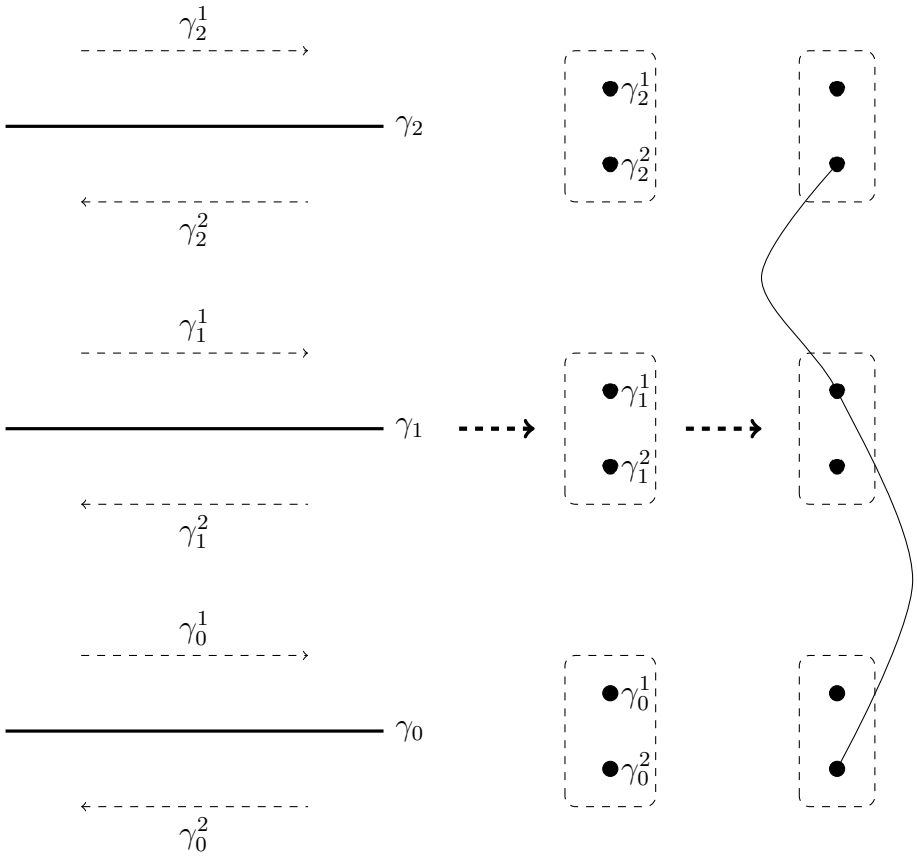

Figure 4.10: Process of converting a set of lines into a graph, followed by a GTSP instance.

4.5 Simulations

The decomposition algorithm was implemented in Python. The algorithm was implemented with the help of computational geometry libraries including Shapely [30] and Visilibity [49]. The heuristic solver GLKH [34] was used as a GTSP solver. Transition costs between segments were computed assuming a Dubins' vehicle model [23].

Our method was compared to two other approaches. The first method relies on an approximate point decomposition of the workspace from [5]. Each point is assigned eight headings. These headings represent possible traversal directions for that point. Finally, the GTSP tour of minimum cost is computed as in [44]. In the second method, the coverage path is computed but the re-optimization procedure is not performed on the initial decomposition. The initial decomposition is generated with a Python library for greedy convex decomposition, based on [26]. We tested all approaches on four different workspaces of similar size but various complexity, based on test environments used in [17].

Performance figures of all three methods are shown in Table 4.1. We compare four aspects of the three approaches: the size of the GTSP instance, the execution time from start to finish, the total length of the tour, and the area covered by the coverage path as percentage of the total area of the polygon. In all cases, our method reduced the number of turns in the final coverage path. This is shown in the reduction in GTSP size, which is equivalent to the reduction in number of straight line segments. Naturally, since our method uses greedy decomposition as a starting point, it runs more slowly then the pure greedy approach. On average, our method is about three times slower than the greedy approach. However, it is faster then the point decomposition by a factor of 100.

Notably, the overall path lengths are shorter with the greedy decomposition compared to our method. However, this is attributed to the greedy decomposition producing many regions whose area is small relative to the size of the footprint. These regions are typically narrow, which makes it difficult to place lines for complete area coverage. Decompositions that produce excessive number of such regions result in a significant portion of the total area uncovered. Our method produces fewer regions and leaves less area uncovered. From Table 4.1, our method leaves on average half of the uncovered area left by the greedy approach. Note that the point decomposition in some cases covers less area than the competing algorithms. However, this occurrence is due to the limitation of our implementation of the approximate decomposition, which does not place any points at a distance of less than r from the boundary, leading to uncovered areas near polygon boundaries. This has the effect of reducing the covered area and the path length as well.

Figure 4.11 shows the resultant paths for three different approaches. The line seg-

Table 4.1: Decomposition methods comparison for multiple test shapes.

		Size	Time	Length	Area
Point Decomposition	Shape 1	7144	2h	312.6	87.8%
	Shape 2	7288	2h	306.3	86.9%
	Shape 3	7464	2h	349.3	88.3%
	Shape 4	6736	2h	312.2	85.1%
Greedy Decomposition	Shape 1	96	**2s**	**231.0**	91.6%
	Shape 2	86	**2s**	**216.8**	84.6%
	Shape 3	136	**6s**	**226.9**	79.8%
	Shape 4	116	**3s**	**226.5**	88.1%
Min Alt Decomposition	Shape 1	**78**	2s	242.3	**95.1%**
	Shape 2	**80**	7s	228.9	**91.7%**
	Shape 3	**92**	15s	228.1	**89.1%**
	Shape 4	**88**	13s	234.4	**93.5%**

ments are orange and Dubins transition costs are green. Note that green segments do not necessarily indicate the path of the robot but rather show the cost associated with the transition. The red lines highlight shared edges of adjacent polygons in the decomposition. The first row of figures show the point decomposition. The second row of figures show the greedy decomposition technique. And the last row shows our re-optimization technique. Figure 4.11 demonstrates a reduction in the number of polygons with our method compared to the greedy decomposition. Inefficiencies in the GTSP tour occur due to the very large size of the GTSP instance, which poses a challenge for the heuristic GLKH solver.

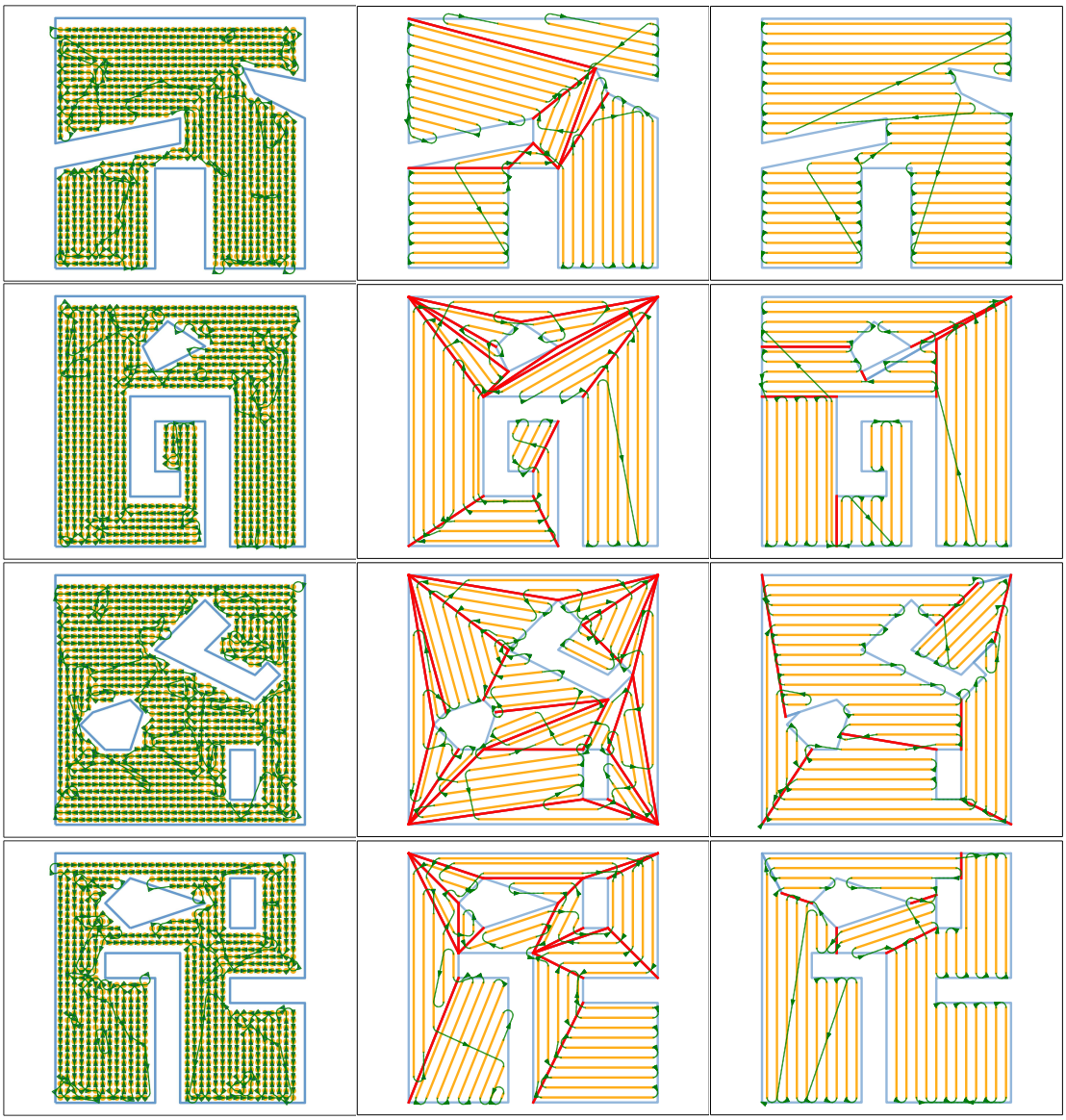

Figure 4.11: First column: point decomposition. Second column: greedy decomposition. Third column: min altitude decomposition.

Chapter 5

Multi-agent Coverage

In this chapter, the multi-agent coverage problem is introduced. The problem statement is formalized in Section 5.1. One of the important aspects of the algorithm is a decomposition metric used during the search algorithm. The design and analysis of this metric is in Section 5.2. The algorithm is proposed in Section 5.3. Section 5.4 presents the computational complexity analysis of the algorithm. Lastly, Section 5.5 presents simulations and a discussion of the results.

5.1 Problem Statement

A multi-agent coverage task is defined over a workspace W and a team of N robots. The workspace W represents the area to be covered. The robots on the team are tasked with coverage. Each robot on a team has the same dynamics and hence, the same free configuration space $Q_{i,\text{free}}$. Each robot also has a coverage map $\mathcal{M}_i(q)$, which maps a configuration q of robot i to a set of points directly under the robot's coverage footprint. The objective of the problem is to compute a path for each robot such that the entire coverable space $\mathbb{C}$ is covered and the maximum path cost amongst all robot is minimized. This objective is motivated by the following.

Complicated tasks can be solved in a distributed manner with a team of robots. This is where the overall task is divided into N subtasks and assigned to N robots in the team. With this setup, it is often undesirable when one robot finishes its assigned subtask much later than other robots. This leads to excessive idle times for the rest of the team and unnecessarily long execution time for the overall task. This issue can be addressed if the

subtask with the longest execution time is redistributed amongst other members of the team such that the duration of the overall task is reduced. In other words, amongst a collection of N subtasks, one would like to modify subtasks such that the longest duration is minimized. As an example, such cost organization was used in [13] for multi-depot vehicle routing problem.

Formally, this problem is introduced in Problem 5.1.1.

Problem 5.1.1 (Multi-agent Coverage Path Planning)**.** *Given a workspace W and a team of N homogeneous robots with the same coverage map, compute $\{p_1, p_2, \ldots, p_n\}$ with the following conditions:*

$$\begin{aligned} & p_i \in P_i, \ i = 1, \ldots, N, \\ & \bigcup_{i=1,\ldots,N} \left(\bigcup_{q \in p_i} \mathcal{M}_i(q) \right) = \mathbb{C}, \\ & \max_{i=1,\ldots,N} \{\mathcal{E}_i(p_i)\} \ \textit{is minimized.} \end{aligned} \tag{5.1}$$

Remark 5.1.1 (Heterogeneous Team). Notice that Problem 5.1.1 is defined for a team of homogeneous robots. A heterogeneous team consists of robots with different dynamics, which leads to coverage spaces that vary from robot to robot. Recall that coverage space determined by the configuration space of robot u is defined in the following way:

$$\mathcal{C}_i = \bigcup_{q \in Q_{i,\text{free}}} \mathcal{M}_i(q). \tag{5.2}$$

If $\mathcal{C}_i \neq \mathcal{C}_j$ then there exists some areas in the workspace that are coverable by only one of the robots. From the coverage planner's perspective, these areas need to be identified since their coverability is conditional on the dynamics of robots. This problem is beyond the scope of this thesis; hence, the team is assumed to consist of homogeneous robots only.

•

Remark 5.1.2 (Path Costs). The path cost $\mathcal{E}(p_i)$ is the same cost described in Section 4.1:

$$\mathcal{E}(p_i) = c_1 \ell(p_i) + c_2 a(p_i). \tag{5.3}$$

Notably, this cost consists of two components: linear and angular. Similar assumption about the dynamics of the robots is made where $c_2 \gg c_1$.

•

Similar to steps taken in Chapter 4.1, the Problem 5.1.1 is modified by reducing the search space for paths to a set of segmented feasible paths. The problem statement for this problem is shown in Problem 5.1.2.

Problem 5.1.2 (Multi-agent Coverage Path Planning with Straight Line Segments)**.** *Given a workspace W and a team of N homogeneous robots with the same coverage map, compute $\{p_1, p_2, \ldots, p_n\}$ with the following conditions:*

$$\begin{aligned} &p_i \in P_{i,segmented},\ i = 1, \ldots, n, \\ &\bigcup_{i=1,\ldots,N} \left(\bigcup_{q \in p_i} \mathcal{M}_i(q) \right) = \mathbb{C}, \\ &\max_{i=1,\ldots,N} \{\mathcal{E}_i(p_i)\} \text{ is minimized.} \end{aligned} \tag{5.4}$$

Finally, we transform Problem 5.1.2 into an equivalent decoupled, two step problem. First, observe that a robot i traversing a path p_i covers a region w_i where

$$w_i = \bigcup_{q \in p_i} \mathcal{M}(q). \tag{5.5}$$

By conditions stated in Problem 5.1.2, a team of robots has to entirely cover the coverable space. As a result, a solution to the problem has to satisfy the following:

$$\bigcup_{i=1,\ldots,n} w_i = \mathbb{C} \tag{5.6}$$

Notice that equation (5.6) suggests some sort of decomposition of $\mathbb{C}$. This observation motivates our decoupled approach to this problem. The first subproblem computes a n-cell decomposition for each agent followed by a single-agent path planning for each cell individually. The evaluation of the decomposition is performed with a metric designed to approximate the coverage path cost. This first subproblem is stated in Problem 5.1.3.

Problem 5.1.3 (Workspace Allocation)**.** *Given a workspace W and N robots with same dynamics, compute a decomposition of $\mathbb{C}$ into $\{w_1, w_2, \ldots, w_N\}$ such that*

$$\begin{aligned} &\bigcup_{i=1,\ldots,N} w_i = \mathbb{C}, \\ &\max_{i=1,\ldots,N} \{\chi_i(w_i, q_i^0)\} \text{ is minimized.} \end{aligned} \tag{5.7}$$

We note that the second subproblem is the problem solved in Chapter 4 and will not be covered in this section. The rest of this chapter focuses on the decomposition problem.

5.2 Decomposition Metric

The solution to Problem 5.1.2 is a set of cells forming a decomposition of the coverable space. The evaluation of decomposition is performed with the metric χ that assigns coverage costs to polygons. Ideally, this metric would be the real cost of a coverage path for a given cell. However, as we have shown in Chapter 4, such metric would be computationally expensive. In this chapter, we develop an approximation of such cost that is easy to compute. This metric is shown in Definition 5.2.1. The rest of the section discusses the design of this metric.

Definition 5.2.1 (Decomposition Metric)**.** Given a polygonal workspace W and a robot with dynamics, the partition metric $\chi : W, Q \to \mathbb{R}$ maps the workspace and the robot dynamics to an approximation of a coverage path cost in the following way:

$$\chi(w_j, q_j^0) = c_1[\mathcal{F}_1(q_j^0, w_j) + \mathcal{F}_2(w_j)] + c_2\mathcal{F}_3(w_j). \tag{5.8}$$

Here, w_j refers to a workspace assigned to robot j and q_j^0 refers to the initial position of robot j, $\mathcal{F}_1, \mathcal{F}_2,$ and $\mathcal{F}_3$ refers to individual terms in the metric.

Remark 5.2.1 (Segmented Coverage Path Structure). The design of the metric χ is motivated by the structure of a coverage path. A coverage path p_i for a robot i consists of three parts. The first part is the segment of the path from the starting location of the robot to its assigned region. The second part consisting of all the straight line segments used in the segmented coverage path. The last part includes all the transition segments connecting the straight segments together as shown in Figure 5.1. Notice that the first two parts could be used as approximations for the linear component cost of the coverage path while the thirst part could be approximating the angular component. The following subsections discuss all three components in detail. •

5.2.1 Initialization and Return Distance

This term is designed to approximate the travel distance of a robot from its starting location to its assigned task. To demonstrate the important of this term, consider the following. A team of N robots have N starting locations, $q_1^0, q_2^0, \ldots, q_N^0$. Depending on the decomposition technique, the starting location of a robot may be some distance away from its assigned cell. Since this travel distance may be significant, it cannot be ignored when computing the cost of a path. For example, assume N robots have equal battery charge and same dynamics. In the scenario where robot i is further to its assigned cell then robot

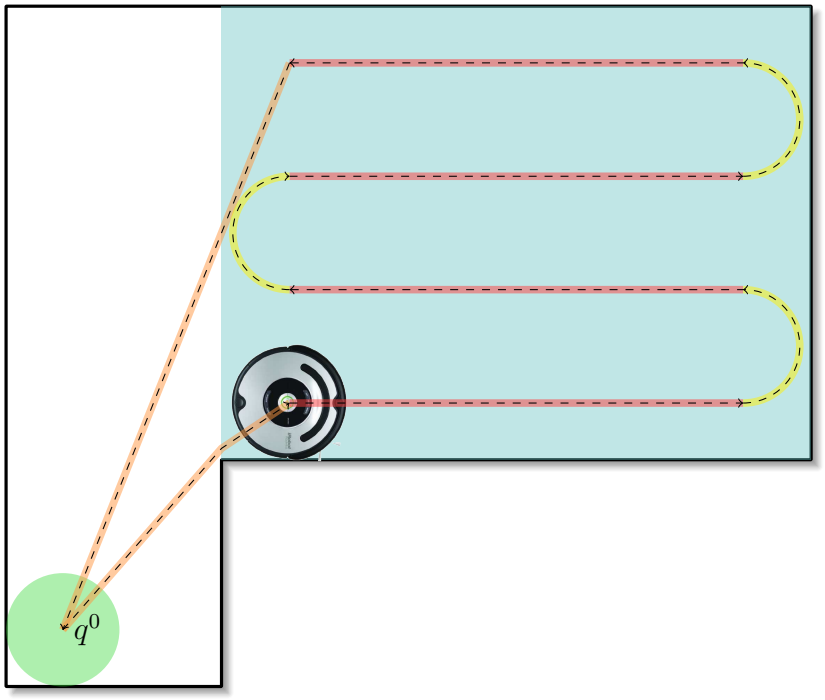

Figure 5.1: The structure of a coverage path. Orange: Segments where the robot travels to its assigned region shown in blue. Red: Path segments that are straight line segments. Yellow: path segments that are transition segments.

j, robot i should be responsible for smaller cell than robot j. We also assume that after the completion of robot's coverage task, it is required to come back to its original starting location.

This first term of the metric χ is the following:

$$\mathcal{F}_1(q_i^0, w_i) = 2 \min_{x \in \partial w_i} ||q_i^0 - x||_2. \tag{5.9}$$

Proposition 5.2.1. *$\mathcal{F}_1$ is a lower bound for the actual distance from the robot's initial position to the start of the coverage path.*

Proof. Denote the combined distance from the starting location of the robot i, q_i^0, to the start of the coverage path in w_i, and the distance from the end of the path to q_i^0 be denoted as

$$\text{dist}(q_i^0, w_i). \tag{5.10}$$

Since a path p_i is defined in a metric space, triangular inequality hold. Hence, the following is true:

$$\mathcal{F}_1(q_i^0, w_i) \leq 2\text{dist}(q_i^0, w_i). \tag{5.11}$$

□

Remark 5.2.2 (Infeasible Paths). The benefit of this approximation is its simplicity and ease of computation. However, it is important to note that a straight path from the starting location to the closest point on the boundary of w_i may not always be feasible as it may intersect an obstacle. Therefore, a situation is possible where the distance from the starting location to the assigned region is close but in reality, requires extensive navigation through obstacles. A way to mitigate this situation is by computing the distance of a shortest feasible path to w_i. However, this requires utilization of visibility graphs, which is computationally expensive [11].

5.2.2 Length of Straight Line Segments

Once the robot reaches the start of its coverage path, the coverage process begins. The second term of the metric χ is designed to approximate the length of all the straight length segments of the coverage path. This approximation is computed using the area of the assigned polygon with a few modifications.

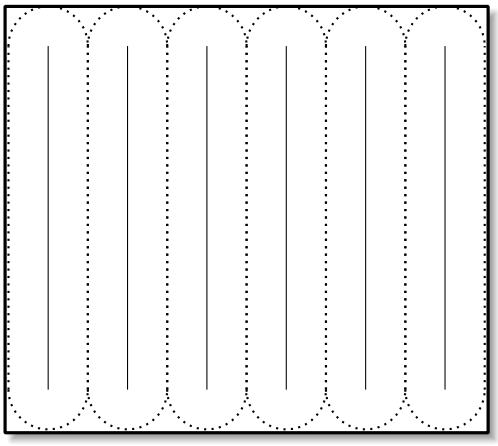

Figure 5.2: Coverage footprints over lines.

The second term of the metric χ is the following:

$$\mathcal{F}_2(w_i) = \frac{A_{\text{actual}}(w_i) - A^{\max}_{\text{uncovered}}(w_i)}{r}. \tag{5.12}$$

In this term, $A_{\text{actual}}(w_i)$ refers to the area of the polygon w_i and $A^{\max}_{\text{uncovered}}(w_i)$ refers to an estimate of all the uncovered area in the polygon.

Suppose a workspace w_i is filled with non-overlapping parallel straight line segments as shown in Figure 5.2. Assuming that the coverage footprint has a width of r, then the area covered by a robot traversing a line of length l_i is $rl_i + k$. Here, k is the amount of area covered beyond the starting and stopping points of a line. These areas are associated with footprints such as a circle as demonstrated by half circles in Figure 5.2. The total area covered in this way is equal to

$$A_{\text{covered}} = \sum_{i=1}^{n} (rl_i + k). \tag{5.13}$$

Note that A_{covered} is not known prior to computing the coverage path. But it can be computed from the following relation:

$$A_{\text{covered}} = A_{\text{actual}} - A_{\text{uncovered}}. \tag{5.14}$$

The term $A_{\text{uncovered}}$ represents the area near the borders of the workspace that remain uncovered as shown in Figure 5.2.

The expression for term $A_{\text{uncovered}}$ is derived in Appendix A and have the following form:

$$A_{\text{uncovered}} = \frac{r^2}{2}(4\csc\theta - \pi). \tag{5.15}$$

However, this term is a function of θ, which is not known prior to computing the actual coverage path. As such, an upper bound is derived in Appendix A:

$$A^{\max}_{\text{uncovered}} = \lceil \frac{\alpha_{\min}}{r} \rceil r^2(4\csc\theta_{\max} - \pi). \tag{5.16}$$

With an easily computable expression for $A^{\max}_{\text{uncovered}}$, the expression for the total length of straight line segments can be stated as follows:

$$\mathcal{F}_2(w_i) = \frac{A_{\text{actual}} - A^{\max}_{\text{uncovered}}}{r}. \tag{5.17}$$

Proposition 5.2.2 (Lower Bound of Total Length of Straight Line Segments)**.** *The term $\mathcal{F}_2(w_i)$ is a lower bound to the actual total length of straight line segments in a segmented path.*

Proof. The term $A^{\max}_{\text{uncovered}}$ was computed with the value of $\theta_{\max}$ that maximizes the $A_{\text{uncovered}}$ function. Moreover, the cardinality of set S is approximated by $\lceil \frac{\alpha_{\min}}{r} \rceil$ where $|S| \leq \lceil \frac{\alpha_{\min}}{r} \rceil$. Hence, the following is true:

$$\begin{aligned} A^{\max}_{\text{uncovered}} &\geq A_{\text{uncovered}}, \\ \implies \mathcal{F}_2(w_i) &\leq \sum_{i=1}^{N} l_i. \end{aligned} \tag{5.18}$$

□

5.2.3 Angular Component

The last term is designed to approximate the angular component of the coverage path. As discussed in Chapter 4, coverage is achieved through segmented paths. With such paths, only transition segments contribute to the angular component of the cost. One can calculate the angular component of the path by computing the path and counting the transition segments. However, this is computationally intensive. In this section, we

develop an approximation for the angular component of the coverage path that makes use of a set of contours. This last term of the metric χ is the following:

$$\mathcal{F}_3(w_i) = 360^o|\mathcal{T}|. \tag{5.19}$$

where $\mathcal{T}$ is a set of contours.

Contours were used for coverage path planning before for computing a contour-parallel paths [33]. A contour is a closed chain generated by *shrinking* the polygon's boundary inwards by a fixed distance. If this process is performed repeatedly where each resultant contour is shrunk by distance $2r$ every iteration, eventually these contours will converge to a point or a line called the center contour as shown in Figure 5.3. Let us denote the set of these closed contours, spaced distance $2r$ apart where r is the radius of the coverage footprint, as $\mathcal{T}$. An example of such a set is shown in Figure 5.3.

The computation of these contours requires no knowledge of the actual coverage path as is solely determined by the shape of the polygon. Furthermore, the angular component of each contour is at least 360^o. This fact and the following proposition leads to the form of $\mathcal{F}_3$ shown in equation 5.19.

Proposition 5.2.3. *Given a polygonal workspace W, suppose a set of contours $\mathcal{T}$ is computed. Let the coverage angle of a Boustrophedon path for W be Φ. Then the following holds:*

$$360^o|\mathcal{T}| \leq \Phi. \tag{5.20}$$

Proof. The result is shown for convex polygons first. The process of iteratively shrinking the boundary converges to the center contour as shown in thick line in Figure 5.3. The distance from any edge of the convex polygon to the center contour is β. Suppose that the minimum altitude of the same convex polygon is α. By construction of the center contour and the minimum altitude, the following holds:

$$\beta \leq \frac{\alpha}{2}. \tag{5.21}$$

However, recall that $\Phi = 180^o\lceil\frac{\alpha}{r}\rceil$ and $|\mathcal{T}| = \lceil\frac{\beta}{r}\rceil$. Assuming that $\frac{\alpha}{r}$ is exact then

$$360^o|\mathcal{T}| \leq 180^o\frac{\alpha}{r}. \tag{5.22}$$

Recall that concave polygons can be decomposed into a set of convex cells. We have also demonstrated that a decomposition can be constructed that minimizes the sum of

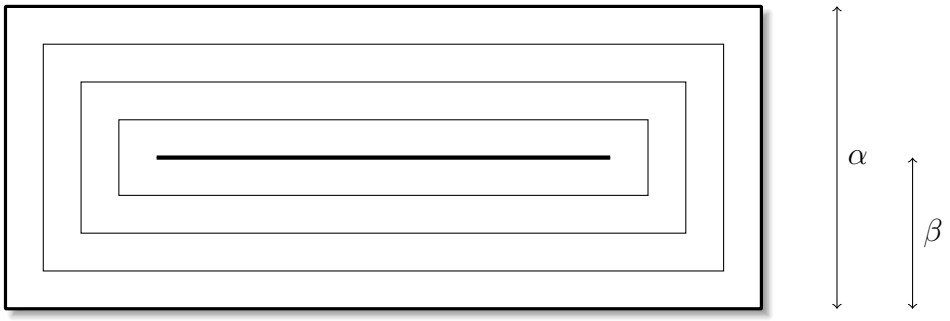

Figure 5.3: Minimum altitude of a convex polygon, α. Distance to the center contour, β.

minimum altitudes over all convex cells in Chapter 4. The sum of all minimum altitudes is related to the number of parallel line segments required for coverage. Hence, a coverage angle can also be computed. Also note that every convex cell has its own center contour. However, it was shown that $360^o \frac{\beta}{r} \leq \Phi$ for convex polygons. Hence, the summation over all cells will hold as well. □

5.3 Area Allocation Algorithm

The algorithm proposed in this section is based on a greedy pair-wise optimization search from some starting point. The procedure is initialized with any n-cell decomposition. For certain adjacent pairs of cells in this decomposition, a search is performed for a new cut that would improve the pair-wise maximum cost. This process terminates when no pair-wise improvements to the decomposition can be made.

The optimization strategy is shown in Algorithm 4. The goal of this algorithm is to identify a cell with the maximum cost and attempt a re-optimization step. As inputs, the algorithm accepts a polygon describing the coverable space and a set of starting positions for each robot in the team. The optimization procedure is initialized with some decomposition on Line 1. This decomposition could be any decomposition that partitions $\mathbb{C}$ into n cells. On Line 3, the adjacency graph is computed for the current decomposition. Costs are computed for all cells in the decomposition and the cell with the maximum cost is identified on Line 4. On Line 5, a recursive procedure is called on the cell with the highest cost. This procedure returns, if possible, a modified decomposition where some two adjacent cells in the decomposition were re-optimized. This procedure terminated when the decomposition remains unchanged after the re-optimization step.

The call on Line 5 of Algorithm 4 is a recursive procedure called on the cell with

Algorithm 4 optimization_procedure($\mathbb{C}, \{q_i^0\}$)

Require: Polygon $\mathbb{C} = \{Z_0, Z_1, \ldots\}$, Set of starting locations $\{q_i^0\}$

1: $\mathcal{D} \leftarrow$ some decomposition of $\mathbb{C}$ consisting of n cells
2: **repeat**
3: $\quad \mathcal{G} \leftarrow$ adjacency graph of $\mathcal{D}$
4: $\quad \chi_i^{\max}, v_i \leftarrow$ the cost and index of highest cost vertex in $\mathcal{G}$
5: $\quad \mathcal{D} \leftarrow$ reopt_recursion$(\mathcal{D}, \mathcal{G}, v_i)$
6: **until** $\mathcal{D}$ stops changing

the highest cost in the current decomposition. This procedure is recursive to account for situations where the cost of the maximum cell cannot be improved via pair-wise re-optimizations with the cell's neighbors. This situations may arise when the costs of the neighbors are similar to the highest cell. As such, the procedure recursively attempts to re-optimize the neighbors through their children with the hope that in the next iteration, the highest cell can be improved. This procedure is described in Algorithm 5.

The Algorithm 5 is a depth-first traversal of the adjacency graph. Cycles are avoided by restricting recursive calls to cells with lower costs. The procedure accepts a decomposition, an adjacency graph, and a vertex representing a cell in the decomposition as inputs. For each neighboring vertex to v_i that has a lower cost, the procedure performs the following steps. On Line 3, a re-optimization cut is attempted for v_i and v_j. If the re-optimization cut was not successful, meaning it failed to find a cut that would decrease the pair-wise maximum, then the same procedure is called on v_j on Line 5. If the re-optimization cut was successful then the current decomposition is modified with the new cut and the procedure terminates.

Algorithm 5 reopt_recursion$(\mathcal{D}, \mathcal{G}, v_i)$

Require: Decomposition $\mathcal{D} = \{C_0, C_1, \ldots\}$, Adjacency graph $\mathcal{G}$, Vertex v_i

1: **for** $v_j \in \mathcal{N}_{v_i}$ **do**
2: $\quad$ **if** $\chi(v_j) < \chi(v_i)$ **then**
3: $\quad\quad \mathcal{D}_{\text{new}} \leftarrow$ reopt_cut$(v_i, v_j, \mathcal{D}, \mathcal{G})$
4: $\quad\quad$ **if** $\mathcal{D}_{\text{new}} = \mathcal{D}$ **then**
5: $\quad\quad\quad \mathcal{D}_{\text{new}} \leftarrow$reopt_recursion$(\mathcal{D}, v_j)$
6: $\quad\quad$ **end if**
7: $\quad\quad$ **return** $\mathcal{D}_{\text{new}}$
8: $\quad$ **end if**
9: **end for**

It should be noted that a breadth-first traversal version of Algorithm 5 is also available and is shown in Algorithm 6. With this approach, the all immediate neighbors of the highest cell are attempted to be re-optimized first before continuing to the next level. It should be noted that in simulations, this approach converged to the same results as the depth-first traversal.

Algorithm 6 reopt_recursion_bft($\mathcal{D}, \mathcal{G}, v_i, Q$)

Require: Decomposition $\mathcal{D} = \{C_0, C_1, \ldots\}$, Adjacency graph $\mathcal{G}$, Vertex v_i, Queue Q

1: **if** Q is empty **then**
2: **return** $\mathcal{D}$
3: **end if**
4: $v_i \leftarrow$ pop from Q
5: **for** $v_j \in \mathcal{N}_{v_i}$ **do**
6: **if** $\chi(v_j) < \chi(v_i)$ **then**
7: $\mathcal{D}_{\text{new}} \leftarrow$ reopt_cut$(v_i, v_j, \mathcal{D}, \mathcal{G})$
8: **if** $\mathcal{D}_{\text{new}} = \mathcal{D}$ **then**
9: Push v_j to Q
10: **else**
11: **return** $\mathcal{D}_{\text{new}}$
12: **end if**
13: **end if**
14: **end for**
15: **return** reopt_recursion_bft$(\mathcal{D}, \mathcal{G}, v_i, Q)$

Another important procedure is described in Algorithm 7. It is a procedure for recomputing a cut that would minimize the maximum cost. In this procedure, the two adjacent cells are combined into one cell on Line 3. A cut space is generated from the origin of the cut on Line 4. This cut space is sampled with equally spaced K points on Line 5. A cut is performed for each point in $\mathcal{L}$ and evaluated. The best cut is found on Line 6. This cut could be the original cut if no improving cuts were found. The cut is then implemented and the current decomposition is modified.

Lastly, an algorithm for computing the metric χ is shown in Algorithm 8. Note that this procedure calculates the three terms as a function of the polygon and the starting location of the robot. The detailed description of the computation of these terms was described in Section 5.2.

An example of the algorithm in action is shown in Figure 5.4. The top picture depicts the initial decomposition of the rectangular workspace into three cells. The initial positions

Algorithm 7 reopt_cut$(v_i, v_j, \mathcal{D}, \mathcal{G})$

Require: Vertices v_i, v_j, Decomposition $\mathcal{D}$, Adjacency graph $\mathcal{G}$

$c \leftarrow$ shared edge between v_i and v_j
$p_s, p_e \leftarrow$ endpoints of c
$w_{\text{temp}} \leftarrow \mathcal{D}(v_i) \cup \mathcal{D}(v_j)$
$S \leftarrow$ cut space in w_{temp} generated from p_s
$\mathcal{L} \leftarrow$ sample S with K points
$p_e^{'} \leftarrow$ point from $\mathcal{L}$ that minimizes the maximum $\{\chi(w_a), \chi(_b)\}$
$c^{'} \leftarrow$ cut $(p_s, p_e^{'})$
$w_a, w_b \leftarrow$cut w_{temp} with $c^{'}$
$\mathcal{D}_{\text{new}} \leftarrow \mathcal{D}$ modified with w_a, w_b
return $\mathcal{D}_{\text{new}}$

Algorithm 8 compute_$\chi(q^0, w)$

Require: Initial position q^0, Polygon $w = \{Z_0, Z_1, \ldots\}$

$\mathcal{F}_1 \leftarrow 2 \min_{x \in \partial w} ||q^0 - x||_2$
$A_{\text{actual}} \leftarrow$ area of w
$\Theta \leftarrow$ angles of all edges w.r.t. the x-axis.
$\theta_{\max} \leftarrow$ largest difference between any two pairs of elements of Θ
$\alpha_{\min} \leftarrow$ minimum altitude of w
$A^{\max}_{\text{uncovered}} \leftarrow \alpha_{\min} r(4 \csc \theta_{\max} - \pi)$
$\mathcal{F}_2 \leftarrow \frac{A_{\text{actual}} - A^{\max}_{\text{uncovered}}}{r}$
$\mathcal{T} \leftarrow$ compute contours of w
$\mathcal{F}_3 \leftarrow 360^o |\mathcal{T}|$
return $c_1(\mathcal{F}_1 + \mathcal{F}_2) + c_2 \mathcal{F}_3$

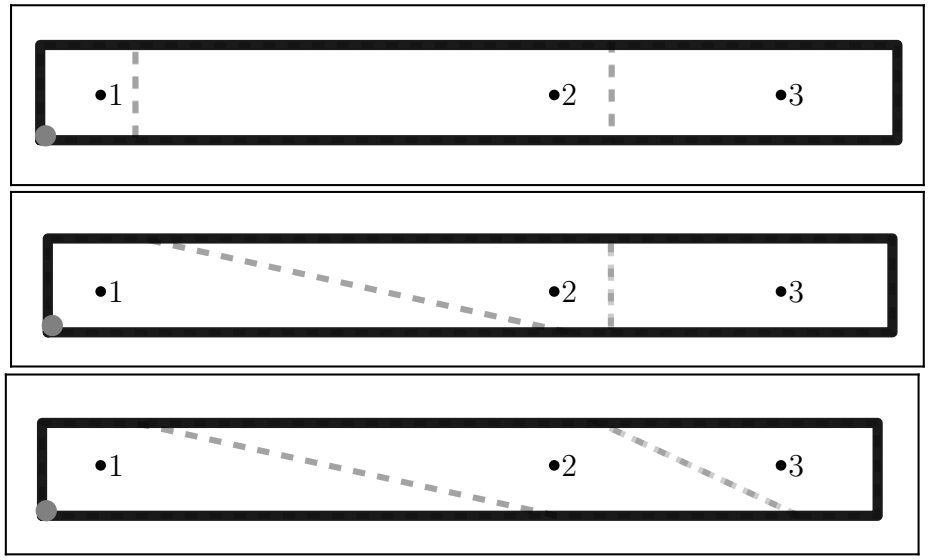

Figure 5.4: Top: initial decomposition. Middle: after first iteration. Bottom: after second iteration.

of the three robots are in the lower left corner of the workspace. The initial decomposition was designed such that the cells two and three has almost equal costs while cell one has the minimum cost. The algorithm identifies the cell three to be the cell with the highest cost. During the recursive call, the algorithm attempts to re-optimize the cut between cell three and two. However, due to costs being almost equal, no improving cuts were found. As a result, a recursive call is called on cell two. Here, cell two is re-optimized with cell one, resulting in lower cost of cell two shown in the middle picture. Finally, the bottom picture shows the next iteration of the algorithm where the cell three is re-optimized with the modified cell two, resulting in the decreased maximum cost.

5.4 Computational Complexity

The run time of Algorithm 8 is determined as follows. Line 1 computes the minimum distance from a fixed points to a set of points. This operation is performed through an exhaustive search in $\mathbb{O}(n)$ time by iterating over all vertices in the set. On Line 2, the area of a polygon is computed in $\mathbb{O}(n)$ time by using Gauss's area formula. On Line 3, a set of angles is computed in $\mathbb{O}(n)$ time by traversing over all edges of the polygon. On Line 4, the largest pair-wise difference between any two pairs of elements is computed in $\mathbb{O}(n^2)$ time

by evaluating all possible pairings. The minimum altitude is computed using Algorithm 1 in $\mathbb{O}(n^2 \log n)$ time. A set of contours is computed on Line 8. In our implementation of this step, a Shapely library is used. The run time of this implementation is not known. However, a robust implementation of contour computation was proposed by Chen [16]. The algorithm computes one contour in $\mathbb{O}((n+k)\log n)$ time where n is he number of edges and k can be roughly approximated by the number of convex vertices in the polygon. Hence, $n + k = 2n$ for convex polygons. Furthermore, to compute the set of contours $\mathcal{T}$, this operation needs to be called several times. This number of times is determined by the shape of the polygon. However, a upper bound on this number can be computed using the minimum altitude of a polygon $\lceil \frac{\alpha_{\min}}{r} \rceil$. Let us refer to this quantity as d. As such, the overall complexity of Algorithm 8 is $\mathbb{O}\left((n+d)n \log n\right)$.

Algorithm 7 computes an improving cut if it exists. Starting with Line 1, an edge is found that is common to both adjacent cells. One way of computing such edge is by checking equality for each pair of edges from both polygons. This operation is $\mathbb{O}(mn)$ time where m and n is the number of verticies in left and right polygons respectively. On Line 3, a union of the two polygons is computed. This operation can be performed by combining the two boundary chains together at the common edge. Since the indices of the shared edge were calculated on Line 1, this operation runs in $\mathbb{O}(1)$ time. On Line 4, a cut space is calculated similarly to Algorithm 3. Efficient $\mathbb{O}(m+n)$ algorithms for computing a visibility polygon exist such as one in [25]. The sampling step on Line 5 computes K samples on the cut space. This operation is simple given a line segment. Hence, this operation has to be performed in $\mathbb{O}(m+n)$ time in the worst case, potentially on every edge of a combined polygon. On Line 6, a point on the cut space is found that results in the best cost cut. This operation is performed by performing cuts at every point in the set and computing the metric for both polygons. Followed by choosing a cut with the best cost. The cut can be performed in $\mathbb{O}(m+n)$ time. The metric χ has to be computed for both polygons with the following runtime:

$$\mathbb{O}\left((n+d_n)n \log n + (m+d_m)m \log m\right). \tag{5.23}$$

It should be noted that there is no relation between n and d_n. Since d_n is related to the dimensions of the polygon whereas n is the number of vertices. Hence, since K is a constant, the whole step has the following run time:

$$\mathbb{O}\left((m+n)\left[(n+d_n)n \log n + (m+d_m)m \log m\right]\right). \tag{5.24}$$

Finally, the decomposition is modified in constant time.

Finally, Algorithm 4 is a recursive re-optimization algorithm. The recursive call in Algorithm 5 can run as many as N times in the worst case where N is the number of

cells in the decomposition. However, it is difficult to determine the number of iterations required for Algorithm 4 to converge. It is clear that this algorithm converges because only the cut that decrease the maximum pair-wise cost are allowed. As such, the whole adjacent graph is traversed looking for possible improvement cuts. If no such cuts were identified then the algorithm terminates.

5.5 Simulations

Our decomposition algorithm was implemented in Python. The algorithm was implemented with the help of computational geometry libraries including Shapely [30] and Visilibity [49]. The heuristic solver GLKH [34] was used as a GTSP solver. Transition costs between segments were computed assuming a Dubins' vehicle model [23].

The approach was tested on its ability to improve the existing decomposition with the goal to minimize the maximum cost. Several test shapes of varying complexity were used in simulations. The initial decomposition of the polygons was performed by hand. In all test runs, four robots were used with starting locations usually located at the corners of the polygon. For Algorithm 7, K is set to 100. To relate the metric χ to the length of the coverage path, actual coverage paths are computed for each robot in the team using Algorithm 2.

Performance figures for the algorithm are shown in Table 5.1 and Table 5.2. We compare the improvements in the maximum cost χ as well as the actual lengths of the coverage paths. This comparison demonstrates that our approach works as intended in minimizing the maximum cost. We also compare the range of costs over four cells in the decomposition. This comparison is to demonstrate the distribution of workloads amongst all robots in the team. The lengths of the coverage paths were computed by solving the single agent coverage problem for each robot in the team using the approach introduced in Chapter 4. We note that the algorithm has improved the maximum value of metric χ on average by 26% and the actual length of the coverage path by 28%, which suggests that the algorithm works as intended. This is confirmed bu the data on the range of costs, which was also reduced by 96% and 84% for metric χ and path lengths respectively. The reduction in the range of costs demonstrates that the resultant costs over four cells is close together. It should be noted that the data from the tables demonstrate strong correlation between the metric χ and the actual coverage path length, which suggests validity of the metric's design.

Figure 5.5 and Figure 5.6 demonstrate decompositions together with the coverage paths. The assignment of paths to the robot is done with color coding. It should be noted that

Table 5.1: Improvement to the maximum cost.

	Max χ			Max Tour Length		
	Old	New	%	Old	New	%
Shape 1	47.5	**41.6**	12.4%	30.7	**26.0**	15.3%
Shape 2	228.1	**155.3**	31.9%	187.6	**115.4**	38.5%
Shape 3	214.3	**154.3**	28.0%	184.6	**125.9**	31.8%
Shape 4	220.4	**157.5**	28.5%	184.1	**132.7**	27.9%
Shape 5	205.0	**145.9**	28.8%	162.9	**117.1**	28.1%

Table 5.2: Improvement to the range of costs.

	Range χ			Range of Tour Lengths		
	Old	New	%	Old	New	%
Shape 1	15.0	**0.3**	98.00%	14.5	**3.4**	76.55%
Shape 2	169.0	**8.3**	95.09%	157.4	**3.0**	98.09%
Shape 3	146.3	**11.3**	92.29%	142.5	**28.5**	79.98%
Shape 4	121.6	**5.9**	95.19%	122.4	**22.6**	81.51%
Shape 5	80.8	**0.02**	99.98%	89.8	**14.5**	83.84%

after re-optimizations, the robots do not necessarily retain their previous color. Initial positions of robots are shown as thick circles located on the boundary of the polygon. The boundary of a cell is shown in a dashed gray line. Furthermore, Algorithm 2 from Chapter 4 may add minimum altitude cuts in each cell shown with red lines. The dotted segments of the path represent transition segments computed using Dubin's vehicle model. Note that these segments do not necessarily indicate the path of a robot but rather show the cost associated with the transition. As such, they may cross the boundary of the polygon. The first column of figures demonstrates the initial decomposition and the second column shows the decomposition after re-optimization.

We should note the limitation of our implementation that arises when cells contain holes. Due to the limitation of the computational geometry library, Algorithm 7 cannot consider a pair of adjacent cells where the union of them contains a hole even if there exists improving cuts. This limitation is evident in the example shown in Figure 5.7. Here, the yellow cell from the second column is never considered for re-optimization because combining it with any of its two adjacent cells results in the presence of a hole. As such, this particular cell is not changed throughout iterations of the algorithm.

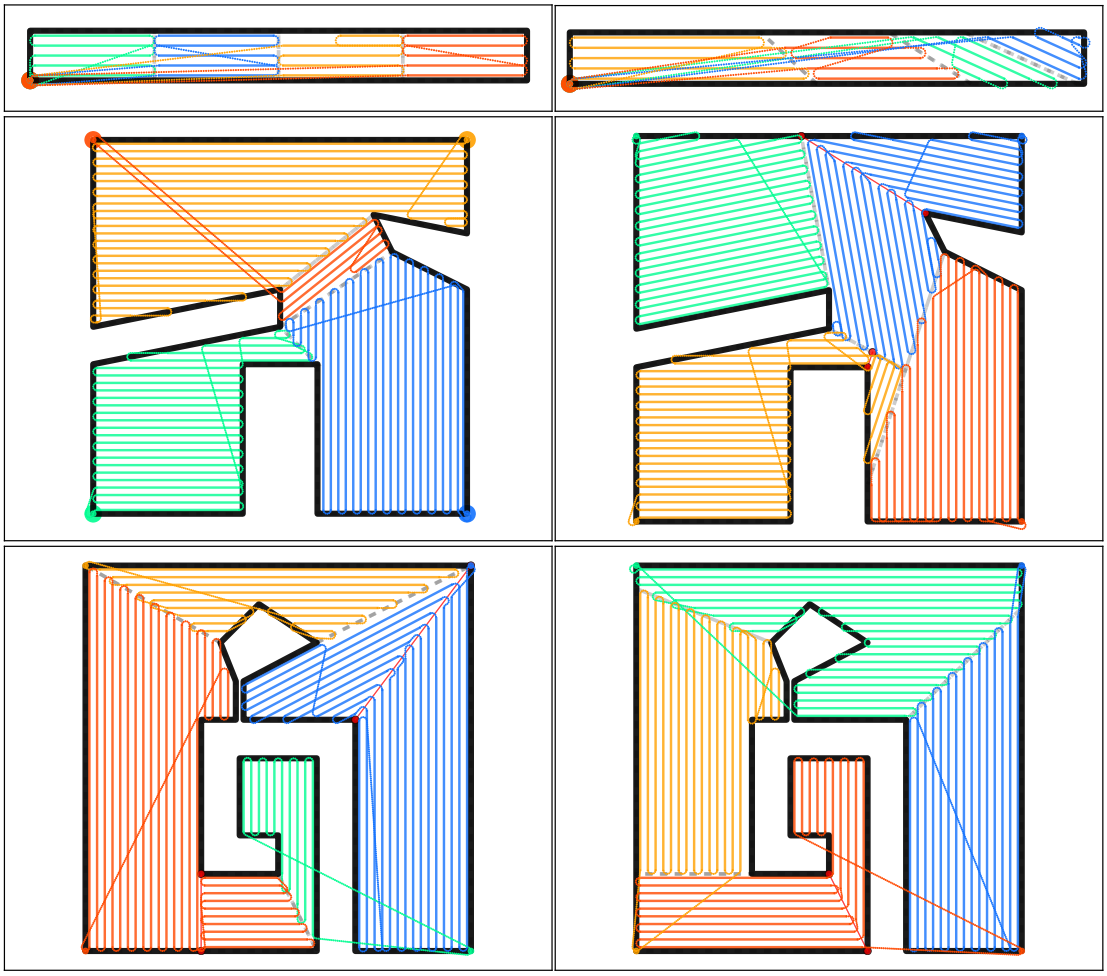

Figure 5.5: Decomposition results for shape 1,2, and 3. First column: decomposition before re-optimization. Second column: after re-optimization.

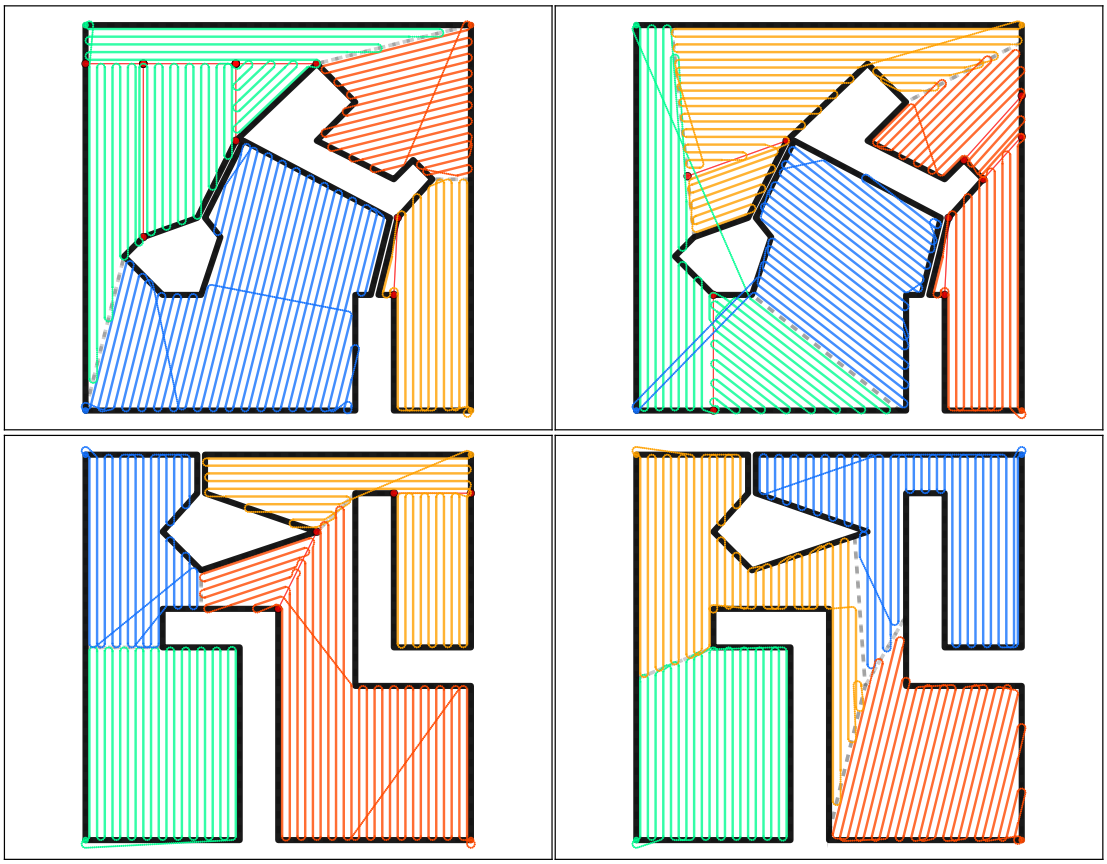

Figure 5.6: Decomposition results for shape 4 and 5. First column: decomposition before re-optimization. Second column: after re-optimization.

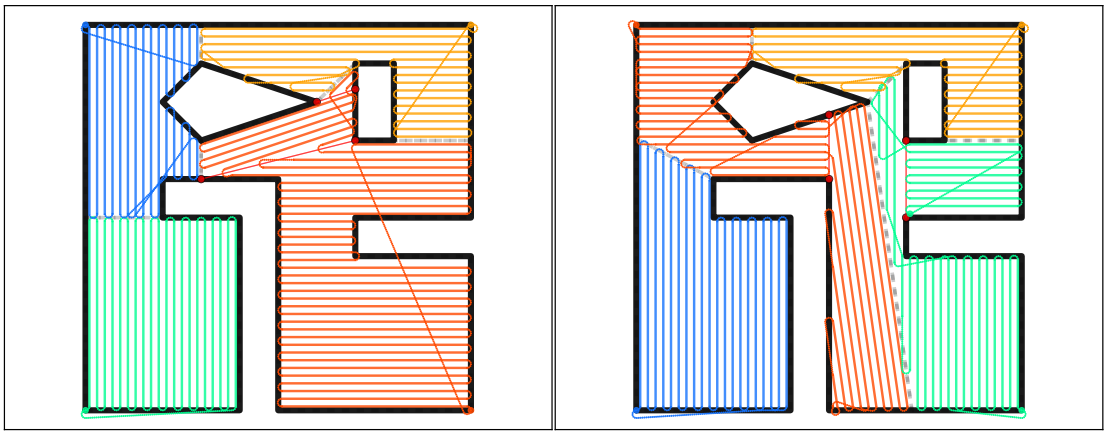

Figure 5.7: Hole Re-optimization results for shape 4 and 5. First column: decomposition before re-optimization. Second column: after re-optimization.

Chapter 6

Conclusions and Future Work

In this thesis, we study two coverage problems. The first problem is a single agent coverage path planning problem with minimum turns. The objective of this problem is to compute a path that covers the entire workspace and has the minimum number of turns. A greedy algorithm is proposed that achieves a local minimal solution via an iterative re-optimizing decomposition.

The second part of the thesis tackles a multi-agent coverage path planning problem. The objective of the problem is to plan a coverage path for a team of robots such that the entire workspace is covered and the maximum coverage cost between all robots in the team is minimized. An approximation for the cost of the path is developed and an efficient algorithm is proposed. This approximation is used to partition the workspace into a set of regions assigned to each robot individually. The remainder of the problem solves a single agent coverage problem for each robot individually.

6.1 Future Work on Single Agent Coverage

There are several potential modifications that would reduce the computational load of the algorithm. The most computational intensive task in Algorithm 3 is computing the cut candidates for each pair of altitude directions (Line 6-17). The computation load can be reduced by only evaluating a subset of direction pairs as follows.

Define $\underline{W}_\ell$ and $\underline{W}_r$ to be the polygons on either side of the cone of bisection. It is important to note that the cone of bisection is not a part of these polygons. We compute look-up tables of altitudes for every direction for $\underline{W}_\ell$ and $\underline{W}_r$ separately such that

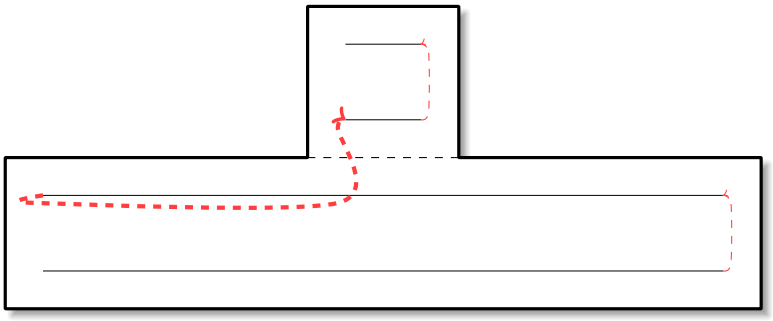

Figure 6.1: An example of a undesired cell transition.

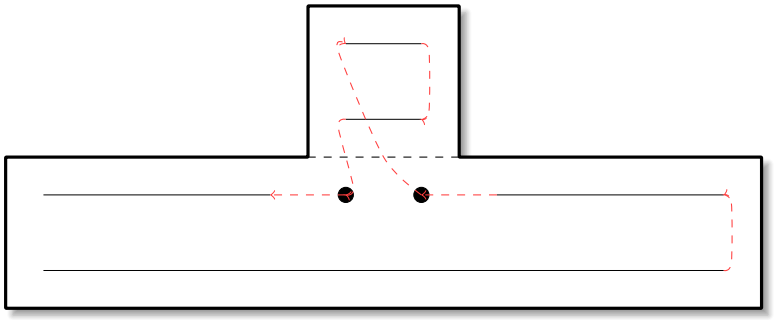

Figure 6.2: An example of a improvement to inter-cell transitions.

$\alpha(\underline{W}_\ell, \text{dir}_1)$ returns the altitude of $\underline{W}_\ell$ in direction dir_1. These look-up tables represent the best case scenario for a pair of directions. In the main loop of Algorithm 3, we keep track of the best altitude so far. A check is added to the main loop such that for a pair of directions, dir_1 and dir_2, if $\alpha(\underline{W}_\ell, \text{dir}_1) + \alpha(\underline{W}_r, \text{dir}_2)$ is larger than the best altitude so far, then this pair of directions cannot be optimal, and no further computation is needed.

Also, the quality of the path produced by the proposed method can be increased via an improved sampling. The paths generated by the proposed method may exhibit some undesirable properties that complicate the transitions between cells. An example of such paths is shown in Figure 6.1, where the thick dashed line represents a transition that is very costly. A better alternative is to resample the long line closest to the adjacent cell to allow for more *elegant* transitions. An example if shown in Figure 6.2. This method avoids long and costly transitions between cells.

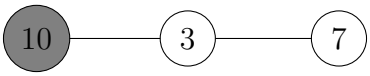

Figure 6.3: An example where the current re-optimization technique terminates.

6.2 Improvements to Multi-Agent Coverage

One of the main mechanisms for re-optimization is the search for an improving pair-wise cut for two polygons. In this work, the search is limited to a cone of bisection for all reflex vertices. There is no guarantee that the search space of this approach contains the optimal. Considering other search techniques could prove to be beneficial in aiding the search for optimal cuts. For example, by also considering cuts outside the cone of bisection, the search space is increased and the chances of finding an optimal cut are improved. Also, better search techniques could be used to improve the computational burden of the algorithm, i.e., binary search. Better techniques could results in few iterations requires to converge to a minimum, thus reducing the computational burden.

The main limitation of Algorithm 4 is the greedy nature of choosing neighbors to the current cell for re-optimization. That is, a neighboring cell is chosen only if it has a lower cost compared to the current cell. This condition creates a possibility of a situation where the proposed algorithm terminates but there are modifications to the decompositions that are still possible. An example is provided in Figure 6.3 where the nodes represent cells, the edges represent adjacency between cells, and the number represent the value of metric χ. Suppose that no cut exists for cells with costs of ten and three. The proposed algorithm will terminate after one iteration. However, it is possible that by making a cut for cells with cost three and seven such that the cost of seven is increases slightly, the changes in the decomposition allows for a new cut between ten and three that reduces the global maximum cost.

References

[1] Ercan U Acar and Howie Choset. Sensor-based coverage of unknown environments: Incremental construction of morse decompositions. *The International Journal of Robotics Research*, 21(4):345–366, 2002.

[2] Ercan U Acar, Howie Choset, Alfred A Rizzi, Prasad N Atkar, and Douglas Hull. Morse decompositions for coverage tasks. *The International Journal of Robotics Research*, 21(4):331–344, 2002.

[3] Ercan U Acar, Howie Choset, Yangang Zhang, and Mark Schervish. Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods. *The International journal of robotics research*, 22(7-8):441–466, 2003.

[4] Esther M Arkin, Michael A Bender, Erik D Demaine, Sándor P Fekete, Joseph SB Mitchell, and Saurabh Sethia. Optimal covering tours with turn costs. *SIAM Journal on Computing*, 35(3):531–566, 2005.

[5] Esther M Arkin, Sándor P Fekete, and Joseph SB Mitchell. Approximation algorithms for lawn mowing and milling. *Computational Geometry*, 17(1):25–50, 2000.

[6] Gökhan M Atınç, Dušan M Stipanović, Petros G Voulgaris, and Mansour Karkoub. Supervised coverage control with guaranteed collision avoidance and proximity maintenance. In *52nd IEEE conference on decision and control*, pages 3463–3468. IEEE, 2013.

[7] Prasad N Atkar, Aaron Greenfield, David C Conner, Howie Choset, and Alfred A Rizzi. Uniform coverage of automotive surface patches. *The International Journal of Robotics Research*, 24(11):883–898, 2005.

[8] T Balch. The case for randomized search. In *Workshop on Sensors and Motion, IEEE International Conference on Robotics and Automation, San Francisco, CA*, pages 213–215, 2000.

[9] Antonio Barrientos, Julian Colorado, Jaime del Cerro, Alexander Martinez, Claudio Rossi, David Sanz, and João Valente. Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots. *Journal of Field Robotics*, 28(5):667–689, 2011.

[10] Timothy Bretl and Seth Hutchinson. Robust coverage by a mobile robot of a planar workspace. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4582–4587. IEEE, 2013.

[11] F. Bullo and S. L. Smith. *Lectures on Robotic Planning and Kinematics*. 2015.

[12] Zuo Llang Cao, Yuyu Huang, and Ernest L Hall. Region filling operations with random obstacle avoidance for mobile robots. *Journal of Robotic systems*, 5(2):87–102, 1988.

[13] John Carlsson, Dongdong Ge, Arjun Subramaniam, Amy Wu, and Yinyu Ye. Solving min-max multi-depot vehicle routing problem. *Lectures on global optimization*, 55:31–46, 2009.

[14] John Gunnar Carlsson. Dividing a territory among several vehicles. *INFORMS Journal on Computing*, 24(4):565–577, 2012.

[15] Christos G Cassandras, Xuchao Lin, and Xuchu Ding. An optimal control approach to the multi-agent persistent monitoring problem. *IEEE Transactions on Automatic Control*, 58(4):947–961, 2013.

[16] Xiaorui Chen and Sara McMains. Polygon offsetting by computing winding numbers. In *ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 565–575. American Society of Mechanical Engineers, 2005.

[17] Young-Ho Choi, Tae-Kyeong Lee, Sang-Hoon Baek, and Se-Young Oh. Online complete coverage path planning for mobile robots based on linked spiral paths using constrained inverse distance transform. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5788–5793, 2009.

[18] Howie Choset. Coverage of known spaces: The boustrophedon cellular decomposition. *Autonomous Robots*, 9(3):247–253, 2000.

[19] Howie Choset and Philippe Pignon. Coverage path planning: The boustrophedon cellular decomposition. In *Field and Service Robotics*, pages 203–209. Springer, 1998.

[20] Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1327–1332. IEEE, 2002.

[21] Arun Das, Michael Diu, Neil Mathew, Christian Scharfenberger, James Servos, Andy Wong, John S Zelek, David A Clausi, and Steven L Waslander. Mapping, planning, and sample detection strategies for autonomous exploration. *Journal of Field Robotics*, 31(1):75–106, 2014.

[22] Wesley M DeBusk. Unmanned aerial vehicle systems for disaster relief: Tornado alley. In *AIAA Infotech@ Aerospace Conference, AIAA-2010-3506, Atlanta, GA*, 2010.

[23] Lester E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3):497–516, 1957.

[24] Joseph W Durham, Ruggero Carli, Paolo Frasca, and Francesco Bullo. Discrete partitioning and coverage control for gossiping robots. *IEEE Transactions on Robotics*, 28(2):364–378, 2012.

[25] Hossam El Gindy and David Avis. A linear algorithm for computing the visibility polygon from a point. *Journal of Algorithms*, 2(2):186–197, 1981.

[26] José Fernández, Boglárka Tóth, Lázaro Cánovas, and Blas Pelegrín. A practical algorithm for decomposing polygonal domains into convex polygons by diagonals. *Top*, 16(2):367–387, 2008.

[27] Eric Frew, Tim McGee, ZuWhan Kim, Xiao Xiao, Stephen Jackson, Michael Morimoto, Sivakumar Rathinam, Jose Padial, and Raja Sengupta. Vision-based road-following using a small autonomous aircraft. In *IEEE Aerospace Conference*, volume 5, pages 3006–3015, 2004.

[28] Yoav Gabriely and Elon Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):77–98, 2001.

[29] Douglas W Gage. Randomized search strategies with imperfect sensors. In *Optical Tools for Manufacturing and Advanced Automation*, pages 270–279. International Society for Optics and Photonics, 1994.

[30] Sean Gillies and Contributors. The Shapely library. `https://github.com/Toblerity/Shapely`, 2013.

[31] Enrique Gonzalez, Maricio Alarcon, Paula Aristizabal, and Carlos Parra. Bsa: A coverage algorithm. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 1679–1684. IEEE, 2003.

[32] Enrique Gonzalez, Oscar Alvarez, Yul Diaz, Carlos Parra, and Cesar Bustacara. Bsa: a complete coverage algorithm. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2040–2044. IEEE, 2005.

[33] Martin Held. *On the computational geometry of pocket machining*, volume 500. Springer Science & Business Media, 1991.

[34] Keld Helsgaun. An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

[35] Keld Helsgaun. General k-opt submoves for the lin–kernighan tsp heuristic. *Mathematical Programming Computation*, 1(2-3):119–163, 2009.

[36] Susan Hert and Vladimir Lumelsky. Polygon area decomposition for multiple-robot workspace division. *International Journal of Computational Geometry & Applications*, 8(04):437–466, 1998.

[37] Susan Hert, Sanjay Tiwari, and Vladimir Lumelsky. A terrain-covering algorithm for an auv. In *Underwater Robots*, pages 17–45. Springer, 1996.

[38] Christian Hofner and Günther Schmidt. Path planning and guidance techniques for an autonomous mobile cleaning robot. In *Intelligent Robots and Systems' 94.'Advanced Robotic Systems and the Real World', IROS'94. Proceedings of the IEEE/RSJ/GI International Conference on*, volume 1, pages 610–617. IEEE, 1994.

[39] Peter F Hokayem, Dusan Stipanovic, and Mark W Spong. Dynamic coverage control with limited communication. In *2007 American Control Conference*, pages 4878–4883. IEEE, 2007.

[40] Wesley H Huang. Optimal line-sweep-based decompositions for coverage algorithms. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 27–32, 2001.

[41] Islam I Hussein and Dusan M Stipanovic. Effective coverage control for mobile sensor networks with guaranteed collision avoidance. *IEEE Transactions on Control Systems Technology*, 15(4):642–657, 2007.

[42] J Mark Keil. Polygon decomposition. *Handbook of Computational Geometry*, 2:491–518, 2000.

[43] Jean-Claude Latombe. Exact cell decomposition. In *Robot Motion Planning*, pages 200–247. Springer, 1991.

[44] Jerome Le Ny, Eric Feron, and Emilio Frazzoli. On the Dubins traveling salesman problem. 57(1):265–270, 2012.

[45] Tae-Kyeong Lee, Sang-Hoon Baek, Young-Ho Choi, and Se-Young Oh. Smooth coverage path planning and control of mobile robots based on high-resolution grid map representation. *Robotics and Autonomous Systems*, 59(10):801–812, 2011.

[46] Xuchao Lin and Christos G Cassandras. An optimal control approach to the multi-agent persistent monitoring problem in two-dimensional spaces. *IEEE Transactions on Automatic Control*, 60(6):1659–1664, 2015.

[47] Ivan Maza and Anibal Ollero. Multiple uav cooperative searching operation using polygon area decomposition and efficient coverage algorithms. In *Distributed Autonomous Robotic Systems 6*, pages 221–230. Springer, 2007.

[48] Charles E Noon and James C Bean. An efficient transformation of the generalized traveling salesman problem. *INFOR: Information Systems and Operational Research*, 31(1):39–44, 1993.

[49] K. J. Obermeyer and Contributors. The VisiLibity library. `http://www.VisiLibity.org`, 2008. R-1.

[50] Timo Oksanen and Arto Visala. Coverage path planning algorithms for agricultural field machines. *Journal of Field Robotics*, 26(8):651–668, 2009.

[51] Mark Ollis and Anthony Stentz. Vision-based perception for an automated harvester. In *Intelligent Robots and Systems, 1997. IROS'97., Proceedings of the 1997 IEEE/RSJ International Conference on*, volume 3, pages 1838–1844. IEEE, 1997.

[52] Dimitra Panagou, Dušan M Stipanović, and Petros G Voulgaris. Vision-based dynamic coverage control for nonholonomic agents. In *53rd IEEE Conference on Decision and Control*, pages 2198–2203. IEEE, 2014.

[53] Morgan Quigley, Michael A Goodrich, Stephen Griffiths, Andrew Eldredge, and Randal W Beard. Target acquisition, localization, and surveillance using a fixed-wing mini-uav and gimbaled camera. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 2600–2605. IEEE, 2005.

[54] Ioannis Rekleitis, Ai Peng New, Edward Samuel Rankin, and Howie Choset. Efficient boustrophedon multi-robot coverage: an algorithmic approach. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):109–142, 2008.

[55] Michael Rososhansky, Fengfeng Jeff Xi, and Yuwen Li. Coverage based tool path planning for automated polishing using contact stress theory. In *IEEE International Conference on Automation Science and Engineering*, pages 592–597, 2010.

[56] Allison Ryan and J Karl Hedrick. A mode-switching path planner for uav-assisted search and rescue. In *Proceedings of the 44th IEEE conference on decision and control*, pages 1471–1476. IEEE, 2005.

[57] A Rydberg, O Hagner, M Söderström, and T Börjesson. Field specific overview of crops using uav (unmanned aerial vehicle). *Precision agriculture*, 7:357–364, 2007.

[58] Weihua Sheng, Ning Xi, Mumin Song, Yifan Chen, and Perry MacNeille. Automated cad-guided robot path planning for spray painting of compound surfaces. In *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 3, pages 1918–1923. IEEE, 2000.

[59] Vikas Shivashankar, Rajiv Jain, Ugur Kuter, and Dana S Nau. Real-time planning for covering an initially-unknown spatial environment. In *FLAIRS Conference*, 2011.

[60] Iddo Shnaps and Elon Rimon. On-line coverage of planar environments by a battery powered autonomous mobile robot. In *Algorithmic Foundations of Robotics XI*, pages 571–589. Springer, 2015.

[61] João Valente, Antonio Barrientos, Jaime del Cerro, Claudio Rossi, Julian Colorado, David Sanz, and Mario Garzón. Multi-robot visual coverage path planning: Geometrical metamorphosis of the workspace through raster graphics based approaches. In *International Conference on Computational Science and Its Applications*, pages 58–73. Springer, 2011.

[62] David P Wagner et al. Approximation algorithms for the minimum bends traveling salesman problem. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 406–421. Springer, 2001.

[63] Matthew Walter, Franz Hover, and John Leonard. Slam for ship hull inspection using exactly sparse extended information filters. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1463–1470. IEEE, 2008.

[64] Sylvia C Wong and Bruce A MacDonald. Complete coverage by mobile robots using slice decomposition based on natural landmarks. In *Pacific Rim International Conference on Artificial Intelligence*, pages 683–692. Springer, 2004.

[65] Fumio Yasutomi, M Yamada, and K Tsukamoto. Cleaning robot control. In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pages 1839–1841. IEEE, 1988.

[66] Xin Yu. *Optimization Approaches for a Dubins Vehicle in Coverage Planning Problem and Traveling Salesman Problems*. PhD thesis, Auburn University, 2015.

[67] Alexander Zelinsky, Ray A Jarvis, JC Byrne, and Shinichi Yuta. Planning paths of complete coverage of an unstructured environment by a mobile robot. In *Proceedings of international conference on advanced robotics*, volume 13, pages 533–538, 1993.

Appendix A

Uncovered Area Term Derivation

Suppose a workspace is to be covered and is populated with some straight line segments as shown in Figure A.1. Note the white uncovered areas near the borders of the polygon. An expression for the area of these regions is derived here. First, observe the uncovered area generated by just one line segment as shown in Figure A.2(a), which depicts the top half of a straight segment.

Note from Figure A.2(b), a region of interest is a trapezoid with two parallel sides a and b, an orthogonal segment of length $2r$, and a segment of the boundary of the polygon. The area of such a trapezoid is

$$A = (a + b)r. \tag{A.1}$$

In this case, r is a constant determined by the width of the coverage footprint. However, a and b are determined by r and θ, which is the angle between that particular section of the boundary and the straight line segment as shown in Figure A.2(b). The expressions for a and b are as follows:

$$\begin{aligned} a &= r \tan \frac{\theta}{2}, \\ b &= r \tan \left(90^o - \frac{\theta}{2} \right). \end{aligned} \tag{A.2}$$

Hence, the trapezoid area can be expressed in terms of r and θ as follows:

$$A = 2r^2 \csc \theta. \tag{A.3}$$

The area of a half circle is

$$\frac{\pi r^2}{2}. \tag{A.4}$$

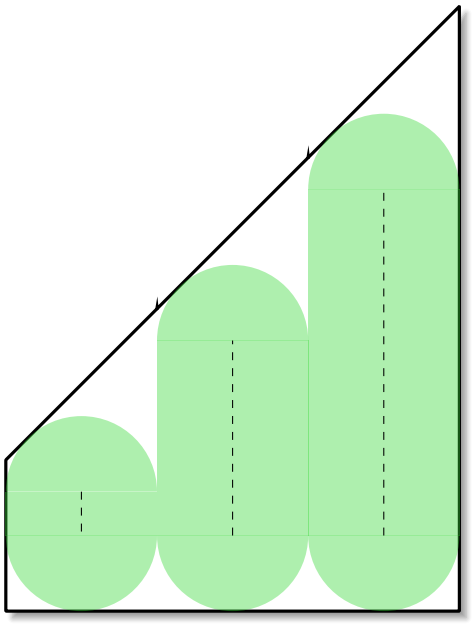

Figure A.1: Sample workspace with parallel line segments.

Therefore, the amount of area that is left uncovered by the top half of the straight line segment is

$$A_{\text{uncovered}} = 2r^2 \csc\theta - \frac{\pi r^2}{2} = \frac{r^2}{2}(4\csc\theta - \pi). \tag{A.5}$$

Note that the expression for $A_{\text{uncovered}}$ is exact and $A_{\text{uncovered}}$ is a function of θ, where θ varies from one edge of a polygon to the next and the orientation of the straight line segments themselves. As such, the computation of this term requires the knowledge of θ. Next, we show an approximation to this value that gets rid of this requirement.

Suppose we compute $\theta_{\max}$ where

$$\theta_{\max} = \arg\max_{\theta\in\Theta} \left(\frac{r^2}{2}(4\csc\theta - \pi) \right). \tag{A.6}$$

Here, Θ is a set of all possible θ that a set of lines can have with the respect to any edge of a polygon. Fortunately, by the result from Huang [40], the optimal orientation of a set of lines is orthogonal to one of the edges of a polygon. Hence, if there are n edges in a polygon then $|\Theta| = n^2$.

Figure A.2: Geometry of the uncovered area.

Hence, the uncovered area can then be approximated as follows:

$$A^{\max}_{\text{uncovered}} = 2|S|\frac{r^2}{2}(4\csc\theta_{\max} - \pi). \tag{A.7}$$

Here, $|S|$ is the number of straight line segments in a workspace scaled by a factor of two to account for both ends of a line. Although $|S|$ is unknown prior to computing the coverage path, it can be approximated by the minimum altitude. This approximation is exact for convex polygons and is an over approximation for non-convex polygons. Therefore, the term is modified as follows:

$$A^{\max}_{\text{uncovered}} = \lceil\frac{\alpha_{\min}}{r}\rceil r^2(4\csc\theta_{\max} - \pi). \tag{A.8}$$