UNIVERSITY OF CALIFORNIA, SAN DIEGO

Visibility-based distributed deployment of robotic teams in polyhedral terrains

A Thesis submitted in partial satisfaction of the requirements for the degree Master of Science

in

Engineering Sciences (Mechanical Engineering)

by

Aaron S. Ma

2016

DEDICATION

I want to dedicate this Thesis to my Mom, Dad, Sister, and baby Niece for always being there for me. To my parents for providing guidance, love, and opportunity that enabled me to go to graduate school. To my Sister for being a great role model in pursuing her career and to my baby niece for being awesome.

TABLE OF CONTENTS

LIST OF SYMBOLS

Symbol	**Description**
$\mathbb{R}$	Set of real numbers
$\mathbb{Z}$	Set of integers
$\mathcal{C}$	Convex set of points
x, y, z	Cartesian coordinates
$\mathcal{G}$	Graph
v	Vertex of graph
V	Vertex set
V^{**}	Vertex set after conversions (see $S^{**}_{2.5}$).
$s_{\overline{p_1 p_2}}$	Slope between v_1 and v_2
e	Edge of graph
E	Edge set
$S_{1.5}$	A 1.5D terrain
$S_{2.5}$	A 2.5D terrain
$S^{*}_{2.5}$	A projection of 2.5D terrain onto surface
$S^{**}_{2.5}$	A reduced $S^{*}_{2.5}$
$S^{**}_{2.5G}$	The resultant connected network of agents in $S_{2.5}$
$\mathcal{J}_{[v_1,v_2]}$	Interval of points between v_1 and v_2
$p_{-}(v)$	Peak immediately left (negative index) of v
$p_{+}(v)$	Peak immediately right (positive index) of v
$\mathcal{P}$	Peak set
K	Interval of points for calculating visibility
Q	Visibility set
a	Agent
A	Agent set
G	Guarding set
G^{ap}	Alternating peak set that agents will guard
v_0	Initially location of all agents
$\mathcal{A}, \mathcal{B}, \mathcal{C}$	Regions that can contain v_0
t	Time step
T	Total time
$\mathcal{R}$	Reducible set
$\mathcal{R}^{*}$	A convex hull in $S^{**}_{2.5}$ where all vertices are visible
$\mathcal{R}_{\text{hull}}$	Boundary of a convex hull in $S^{**}_{2.5}$ where all vertices are visible
Δ	A set of triangular faces

p	Non-redundant pair
P	Set of non-redundant pairs
Γ	Set of all nun-redundant vertices
U	Unoccupied vertices with no unoccupied neighbors
$\mathcal{U}$	Set of undetected vertices
$\mathcal{K}$	Set of vertices with no undetected neighbors
$\mathcal{D}$	Set of vertices with atleast one undetected neighbor
D	Set of vertices that are detected but not colored
COL	Set of colors(labels)
$1, 2, 3, 4, 5, 6$	Labels for coloring $S_{2.5}^{**}$
G_c	Label set that agents will guard
f	Number of faces in a planar graph
Ψ	Max out-degree of vertices in a planar graph

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank Northrop Grumman for their financial support that enabled me focus on my goals and research. I want to thank Professor Jorge Cortés and Sonia Martínez for giving me countless opportunities in my graduate endeavors. It has been an amazing experience so far. I want to thank my graduate student mentors, Dr. Mike Ouimet and Evan Gravelle for all of their help.

Also Professor Mark Sherwin, Dave Enyeart, and Jerry Ramian for their mentorship at UCSB. Finally, I would like to thank my friends Travis Trinh and Eric Nguyen, and my family for making my engineering experience special.

Chapters 1, 2, 3, and 4, in part, have been submitted for publication of the material as it may appear in the Dynamic Systems and Control Conference, 2016, Cortés, Jorge. The thesis author was the primary investigator and author of this paper.

VITA

2012	B. S. in Engineering Sciences (Mechanical Engineering), University of California, Santa Barbara
2012-2014	Development Engineer, Institute for Terahertz Science and Technology
2016	M. S. in Engineering Sciences (Mechanical Engineering), University of California, San Diego

PUBLICATIONS

A. Ma "Visibility-based distributed deployment of robotic teams in polyhedral terrains", Proceedings of the ASME Dynamic Systems and Control Conference, Minneapolis, Minnesota, 2016, DSCC2016-9820, submitted

ABSTRACT OF THE THESIS

Visibility-based distributed deployment
of robotic teams in polyhedral terrains

by

Aaron S. Ma

Master of Science in Engineering Sciences (Mechanical Engineering)

University of California, San Diego, 2016

Professor Jorge Cortés, Chair

Due to recent technological advances, robotic swarms are currently a large interest for surveillance, disaster response, and exploration. In order to solve this problem, we develop distributed deployment strategies for 1.5D and 2.5D polyhedral terrains that are influenced by research in computational geometry, graph theory, and distributed controls. Similarly to the guarding of art gallery problems, we guard an environment through the collective visibility of a team of robots. We consider scenarios where the robots are constrained to moving on the ground in 1.5D and 2.5D polyhedral terrains. Our objective is to determine strategies for deploying robots in polyhedral terrains that guarantees complete visibility of the terrain.

In the 1.5D polyhedral terrain, we determine a set of locations, that guarantees that the terrain is completely visible when occupied. We then develop a set of instructions that each robot distributively executes in order to occupy the set of locations. Finally, we find a closed-form expression for the time required for the 1.5D deployment strategy to complete that scales with the size of the terrain.

In the 2.5D polyhedral terrain, we develop a set of instructions for the robots to follow that collectively explores, colors, and guards the polyhedral terrain. We define rules for the agents to label certain locations that must be occupied in order to achieve complete visibility, inspired by coloring of planar graphs. Finally, we characterize the best and worse time complexity for the algorithm to complete that depends on the structure and size of the polyhedral terrain.

Chapter 1

Introduction

Currently there is large interest in distributed robotics and automation for use in surveillance and disaster response. Similarly to the guarding of art gallery problems, we guard the environment through the collective visibility of a team of robots. Here we consider scenarios where the robots are constrained to moving on the ground in 1.5D and 2.5D polyhedral terrains. Our objective is to determine strategies for deploying robots in polyhedral terrains that are guaranteed to achieve complete visibility within some determined time.

1.1 Literature Review

This thesis builds on research of the classical art-gallery problem [1] in computational geometry. The work [2] shows that $n/3$ guards are sufficient and sometimes necessary to guard the inside of any polygon with n vertices. Many variations of the art-gallery problem exist. We focus here on guarding polyhedral environments. In [3], methods for calculating and analyzing the visibility of polyhedral terrains are explored. [4] discusses a polynomial time approximation scheme for guarding of 1.5-dimensional terrains. A centralized, locally optimal, polynomial-time approximation scheme (PTAS) for guarding a terrain is introduced in [5]. In our analysis and algorithm design for guarding 2.5D polyhedral terrains, we use results from 4 coloring of planar graphs [6]. [7] characterizes the number of agents required to guard a 2.5D terrain using coloring techniques on planar graphs. A planar graph is considered colored when its vertices are labeled in such a way that no two neighbors share the same label. Algorithms have been proposed to color such planar graphs in no more than 5 colors [8, 9] (in time linear with the number of graph vertices), and no more than 4 colors [10] (in time quadratic with the number of graph vertices). While guarding 2.5D polyhedral terrains, we construct a face-spanning tree, a tree that visits every face of a planar graph. Although research in face-spanning trees is sparse, [11] provides results on the lower bound of the number of vertices in a face-spanning tree which we use for our results. The thesis also builds on notions from distributed robotic networks [12] and distributed deployment of mobile robots to guard art galleries based on visibility and line of sight [13].

1.2 Statement of contributions

We design distributed algorithms for robotic teams to achieve full visibility of polyhedral terrains. Our contributions are structured in two blocks, one corresponding to 1.5D environments and the other one corresponding to 2.5D environments.

For 1.5D environments, we begin by characterizing a guarding set to achieve full visibility of the terrain based on identifying alternate peaks. This allows us to determine a number of agents that are always sufficient and some times necessary to guard any 1.5D environment. Building on this result, we design two deployment strategies and determine closed-form expressions for the time it takes each strategy to complete. The first strategy allows for more flexible initial conditions, while the second strategy that we introduce completes in less time. The strategies for deployment in 1.5D environment are iterative processes where the agents communicate through vision, compute, move, and detect where they are in their environment.

In the 2.5D environment we define locations that are redundant in terms of guarding and visibility. We determine the maximum number of locations that are not redundant (removing an agent that guards a redundant location does not change the collective visibility) and use this result as the sufficient number of agents to guard any 2.5D terrain. We determine a distributed 2.5D deployment strategy that yields complete visibility by utilizing planar graph coloring and redundant locations. Finally we find the time that it takes for our 2.5D deployment strategy to complete. Various simulations throughout the thesis illustrate our results.

Chapter 1, in part, has been submitted for publication of the material as it may appear in the Dynamic Systems and Control Conference, 2016, Cortés, Jorge. The thesis author was the primary investigator and author of this paper.

Chapter 2

Overview of terrain guarding

In this chapter, we define material necessary for this thesis as well as our objective and motivation for developing algorithms for polyhedral terrain deployment. We begin by defining notations, planar graphs, coloring of planar graphs, and polyhedral terrains. We end the chapter by stating our objectives and intended contributions in computational geometry and distributed robotics.

2.1 Notation

Given a set S, we let $|S|$ denote its cardinality. We let $\mathrm{ceil} : \mathbb{R} \to \mathbb{Z}$ denote the ceiling function which rounds its argument to the next highest integer. We denote by $\overline{p_1 p_2}$ the line segment between points $p_1, p_2 \in \mathbb{R}^d$. A set $\mathcal{C} \subset \mathbb{R}^d$ is convex if the line segment between any pair of its points is contained in $\mathcal{C}$. In $\mathbb{R}^3$, we use x^p, y^p, and z^p to denote the components of the point $p \in \mathbb{R}^3$. Given $p_1, p_2 \in \mathbb{R}^3$, the slope of $\overline{p_1 p_2}$ is

$$s_{\overline{p_1 p_2}} = \frac{z^{p_2} - z^{p_1}}{\sqrt{(x^{p_2} - x^{p_1})^2 + (y^{p_2} - y^{p_1})^2}}.$$

When convenient, we embed the Euclidean plane $\mathbb{R}^2$ into the Euclidean space $\mathbb{R}^3$ through the map i defined by $i(a_1, a_2) = (a_1, 0, a_2)$. With this embedding, we have $y^v = 0$ for any point $p \in i(\mathbb{R}^2) \equiv \mathbb{R}^2$.

2.2 Planar graphs and coloring

An undirected graph $\mathcal{G} = (V, E)$ is a pair composed of a vertex set V and an edge set E consisting of bidirectional edges between vertices. The degree of a vertex is the number of edges connected to it. Planar graphs are undirected graphs whose vertices belong to $\mathbb{R}^2$ and whose edges can be drawn on a plane in such a way that no edges cross each other.

A planar graph is *colored* when its vertices are labeled in such a way that no two neighboring vertices share the same label. Planar graphs can be colored with no more than 4 colors, cf. [6]. Centralized algorithms can color planar graphs with no more than 4 colors (in $O(n^2)$ time), see [10], and with no more than 5 colors (in $O(n)$ time), see [8]. An example of a 4 colored planar graph is given in Figure 2.2.

2.3 Polyhedral terrains in 1.5D and 2.5D

1.5D and 2.5D polyhedral terrains correspond to the graphs of continuous piecewise affine functions on $\mathbb{R}$ and $\mathbb{R}^2$, respectively. Formally, given a continuous piecewise affine function $f : I \subset \mathbb{R} \to \mathbb{R}$, with I an interval, its associated 1.5D terrain is $S_{1.5}(f) = \{(x, f(x)) : x \in I\} \subset \mathbb{R}^2$. Similarly, given a continuous piecewise affine function $f : I \subset \mathbb{R}^2 \to \mathbb{R}$, with I a polygon, its associated 2.5D terrain is $S_{2.5}(f) = \{(x, y, f(x, y)) : (x, y) \in I\} \subset \mathbb{R}^3$. When convenient, we drop the dependence on f

and simply denote $S_{d.5} \subset \mathbb{R}^{d+1}$ to refer to either of these two cases.

Note that a polyhedral terrain $S_{d.5}$ can also be seen as an undirected graph with vertices in $\mathbb{R}^{d+1}$. In the case $d = 1$, these vertices correspond to the points in $\mathbb{R}^2$ where the graph of two affine components of f intersect. In the case $d = 2$, these vertices correspond to the points in $\mathbb{R}^3$ where the graph of three affine components of f intersect. The set of edges connecting vertices in $S_{1.5}$ and $S_{2.5}$ are denoted $E_{1.5}$ and $E_{2.5}$, respectively. All vertices, v_i in $S_{1.5}$, except for v_1 and $v_{|V|}$ have degree of 2, with neighbors, v_{i-1} and v_{i+1}. It follows that $v \in S_{1.5}$ are ordered monotonically with respect to the x-axis, such that $x_{i-1}^v < x_i^v < x_{i+1}^v$ for $i \in \{2, \ldots, |V| - 1\}$. In $S_{1.5}$, we define $\mathcal{J}_{(v_1,v_2)}$ (resp. $\mathcal{J}_{[v_1,v_2]}$) to be the set of all vertices $v \in V$ such that $\min(x^{v_1}, x_2^v) < x^v < \max(x_1^v, x_2^v)$ (resp. $\min(x_1^v, x_2^v) \leq x^v \leq \max(x_1^v, x_2^v)$). A vertex v_i in $S_{1.5}$ is a *peak* if $s_{v_{i-1},v_i} > s_{v_i,v_{i+1}}$. Conversely v_i is a *valley* if it is not a peak. We denote by $\mathcal{P}$ and $\mathcal{V}$ the collection of peaks and valleys, respectively, in increasing order with respect to their x-coordinate. Given a vertex v, we denote its adjacent peak to the right by $p_+(v)$ and to the left by $p_-(v)$.

In our treatment of 2.5D terrains, we find it convenient to use triangulated planar graphs. Denote by $\mathrm{pr} : \mathbb{R}^3 \to \mathbb{R}^2$ the projection map onto the first two components, $\mathrm{pr}(x, y, z) = (x, y)$. This map projects $S_{2.5}$ onto a planar graph, which denote by $S_{2.5}^*$. Edges maintain their connectivity through conversion, and we denote them by $e^* \in E^*$. Figure 2.2 shows a 2.5D terrain, $S_{2.5}$, transformed into its planar graph equivalent, $S_{2.5}^*$.

Two vertices, v_1 and v_2, are visible to each other if $\overline{v_1 v_2}$ never intersects with $S_{d.5}$. We use the following test to determine if two vertices are visible. Given vertices

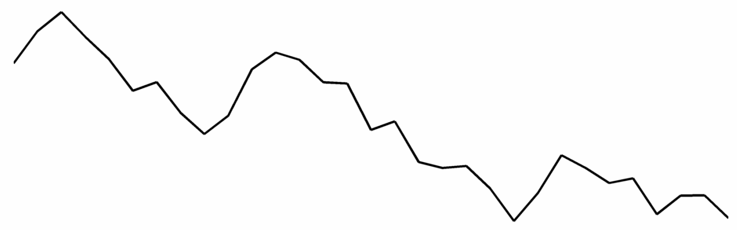
v_1, v_2, the *visibility test* consists of checking whether

$$s_{\overline{v_1 v_2}} > s_{\overline{v_1 w}}, \quad \forall\, w \in K_{v_1 v_2}. \tag{2.1}$$

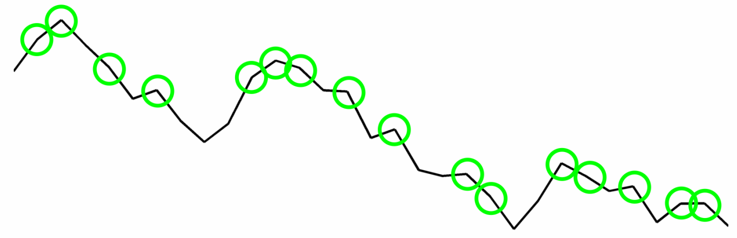
If the test is passed, then the vertices are visible to each other. In $S_{1.5}$, $K_{v_1 v_2} = \mathcal{J}_{(v_1, v_2)}$ is the set of vertices between v_1 and v_2. In $S_{2.5}$, $K_{v_1 v_2}$ is the set of points on $S_{2.5}$ that share x and y-coordinates with $\overline{v_1 v_2}$. The visibility set of a vertex v in $S_{d.5}$, denoted $Q(v)$, is the set of all vertices visible to v. Given $V_w \subset V$, the collective visibility set,

$$Q(V_w) = \bigcup_{v \in V_w} Q(v),$$

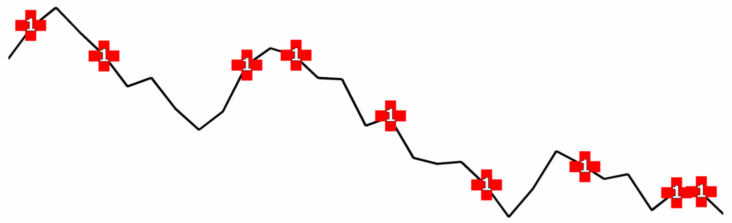
is the set of all vertices visible to them. $S_{d.5}$ is completely visible from V_w if $Q(V_w) = V$. We say that a terrain is fully guarded when an agents occupy $v \in V_w$ such that $Q(V_w) = V$. Figure 2.1 shows $S_{1.5}$, $S_{1.5}$ with *peaks* highlighted, and $S_{1.5}$ with agents that occupy a V_w such that the terrain is fully guarded.



(a) A polyhedral terrain

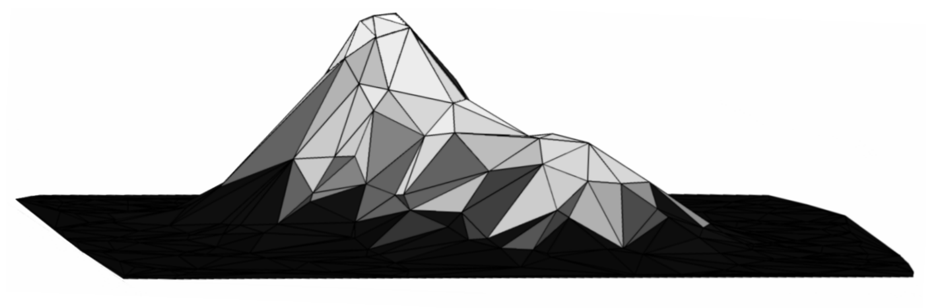


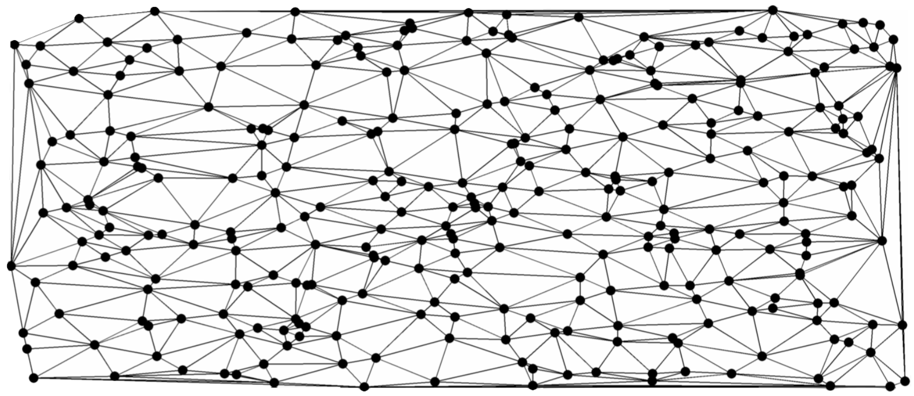
(b) A polyhedral terrain with peaks highlighted



(c) A fully guarded $S_{1.5}$

Figure 2.1: A 1.5D terrain, $S_{1.5}$, is shown by itself, with peaks denoted by green circles, and with $v \in V_w$ such that $Q(V_w) = V$, denoted by red crosses.


(a) A polyhedral terrain, $S_{2.5}$.


(b) The planar graph associated with $S_{2.5}$.

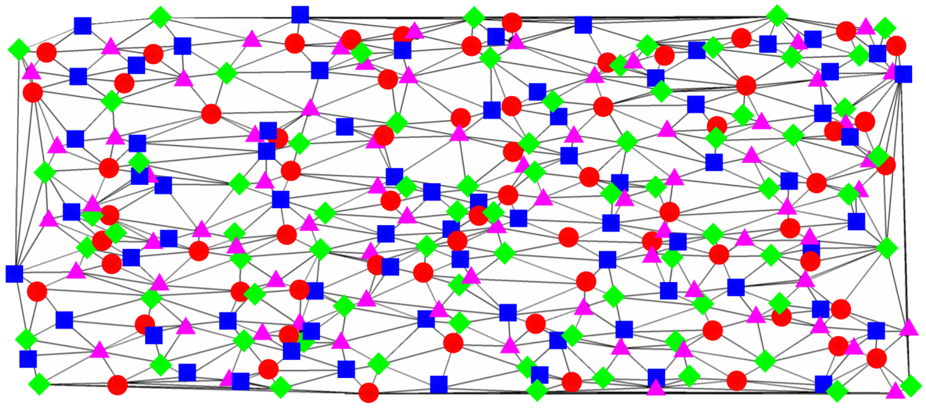

(c) The colored planar graph associated with $S_{2.5}$.

Figure 2.2: The 2.5D terrain converted into a planar graph. The planar graph is then 4-colored.

2.4 The terrain guarding objective

We consider scenarios where a team of robots, deployed on $S_{d.5}$, with $d \in \{1, 2\}$, seek to achieve full visibility of the environment. In this chapter we describe in detail the model for the robotic network and its capabilities. Each individual robot, also referred to as agent, has a unique identifier $i \in \{1, \ldots, |A|\}$, which provides a sense of priority when two agents decide to execute conflicting actions. The agents are capable of omni-directional vision and can localize vertices at infinite distance. Agents are only able to communicate and share information when they are visible to each other. The agents have the capability to share attributes about vertices such as their coordinates and color assignments. In $S_{2.5}$, we allow agents to place relays on a vertex to allow communication between any two agents that occupy vertices that are neighbors of it. In $S_{1.5}$, agents are able to traverse between adjacent peaks at every time step. In $S_{2.5}$, agents are able to traverse between vertices connected by an edge at every time step. We consider the motion of the robots slow in comparison to the time required for computation.

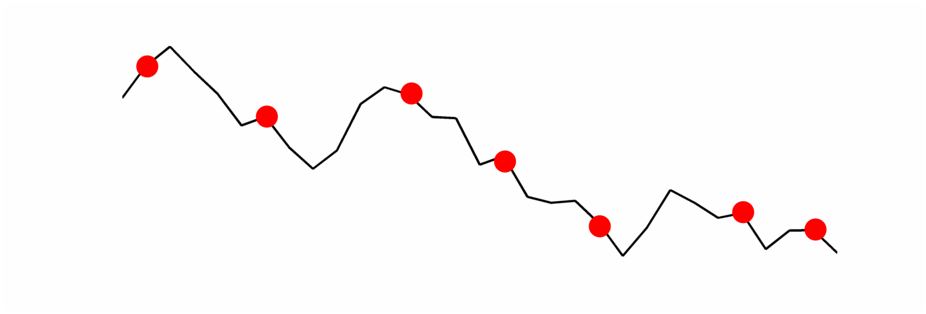
We refer to the guarding set, $G \subset V$, as the set of vertices that the agents decide to occupy. This set is determined in a dynamic fashion by the agents as they explore the environment. $S_{d.5}$ is fully guarded if $Q(G) = V$. Our objective is to design a coordinated strategy for the agents to distributively explore the polyhedral terrain $S_{d.5}$ and determine the guarding set to achieve full visibility. We also seek to characterize the number of agents required to achieve this as well as the time required by the coordination strategies for achieving full visibility.

2.5 Centralized vs. decentralized

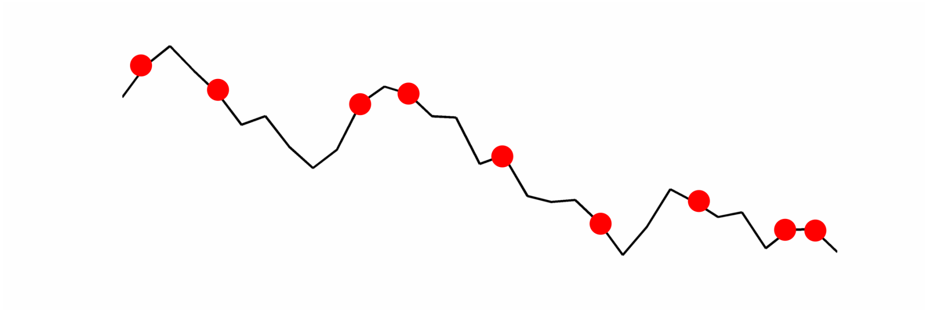
The relevance of robotic swarms and their applications demand novel decentralized algorithms. There are both advantages and disadvantages of a decentralized methods in robotics. Advantages include the parallel computation, the ability for individual agents to allocate more resources to a single task, and robust dynamics. In the terrain guarding case, decentralized methods usually leads to less optimal results.

Many algorithms have been employed to solve the terrain guarding problem in a centralized fashion, and is sometimes required to optimize deployment time or minimize number of robots. Figure 2.3 shows the results of $S_{1.5}$ being solved in a centralized manner that needs less agents than our proposed method. It is particularly difficult to solve the terrain guarding problem from a decentralized standpoint because we need to consider the communication ability of the agents, and that the agents are unaware of the structure of $S_{1.5}$ or $S_{2.5}$ before attempting to guard it. In Figure 2.4 the agents are unaware of the rest of $S_{d.5}$ during execution of the strategies, showcasing the difficulty of optimal deployment.

Chapter 2, in part, has been submitted for publication of the material as it may appear in the Dynamic Systems and Control Conference, 2016, Cortés, Jorge. The thesis author was the primary investigator and author of this paper.

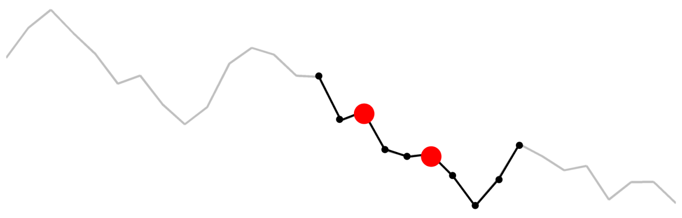


(a) A centralized solution

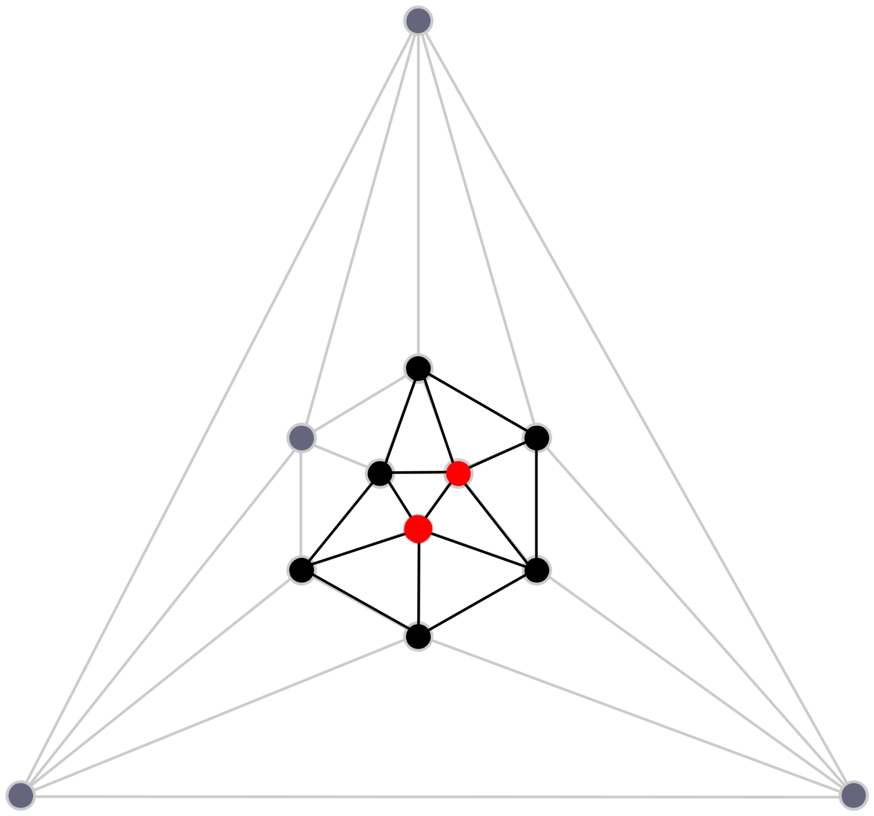


(b) Our decentralized solution

Figure 2.3: The results of a centralized and our decentralized deployment onto $S_{1.5}$



(a) Exploration of $S_{1.5}$



(b) Exploration of $S^{**}_{2.5}$

Figure 2.4: $S_{1.5}$ and $S^{*}_{2.5}$ are shown in the middle of the distributed strategy executions. We show how the agents are unaware of the parts of $S_{d.5}$ that they are unable to see. Vertices and edges are faded out to represent the what information is available to the agents and red dots to represent a vertex occupied by an agent.

Chapter 3

1.5D terrain guarding

This chapter studies the distributed deployment problem over 1.5D polyhedral terrains. We identify a guarding set that guarantees full visibility and study its size to obtain a characterization of a sufficient and sometimes necessary number of robotic agents required to complete the task. Building on this characterization, we design strategies to place agents in the identified guarding set.

3.1 Guarding set via alternate peaks

Here we characterize a guarding set that achieves full visibility of a 1.5D polyhedral terrain $S_{1.5}$. We begin our analysis with a simple fact about the visibility regions of adjacent peaks.

Lemma 3.1.1 *(Visibility from adjacent peaks): Given two adjacent peaks, v_1 and $v_2 \in \mathcal{P}$, all intermediate vertices of $S_{1.5}$ are visible to them, i.e., $\mathcal{J}_{[v_1,v_2]} \subset Q(v_1)$ and $\mathcal{J}_{[v_1,v_2]} \subset Q(v_2)$.*

Proof 3.1.2 *Since v_1 and v_2 are adjacent peaks, all vertices $v_i \in \mathcal{J}_{(v_1,v_2)}$ are valleys* *and have the property,* $s_{v_{i-1},v_i} \leq s_{v_i,v_{i+1}}$*. Therefore the slope between adjacent vertices,* v_i *and* $v_{i+1} \in \mathcal{J}_{(v_1,v_2)}$*, monotonically increases with increasing* x *along the interval* x^{v_1} *to* x^{v_2}*, implying* $s_{v_1 v} > s_{v_1 k}$ *for all* $v \in \mathcal{J}_{(v_1,v_2)}$ *and* $k \in K_{v_1 v} = \mathcal{J}_{(v_1,v)}$*. Hence, all vertices between* v_1 *and* v_2 *are visible from either* v_1 *or* v_2*.* $\square$

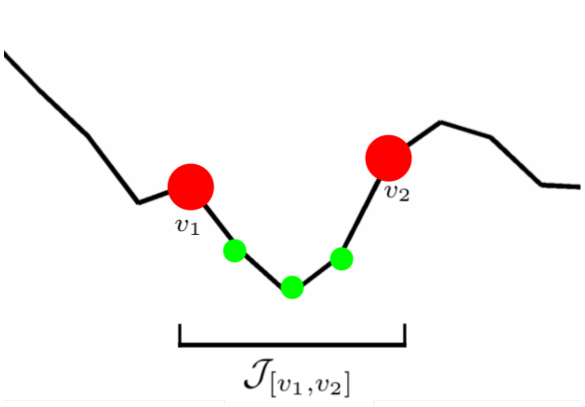


Figure 3.1: Visibility of two adjacent peaks are shown. The red dots represent a vertex occupied by an agent (v_1 and v_2), and the green dots represent visible vertices in between v_1 and v_2

An example of vertices visible as a result of Lemma 3.1.1 is shown in Figure 3.1. As a consequence of Lemma 3.1.1, we deduce that $\mathcal{J}_{[p_-(v),p_+(v)]}$ is visible from v. Inspired by this observation, we consider the subset of alternating peaks, denoted $G^{ap} \subset \mathcal{P}$, corresponding to all peaks with odd indices. Note that if $|\mathcal{P}|$ is even, then one can alternatively consider the set of peaks with even indices.

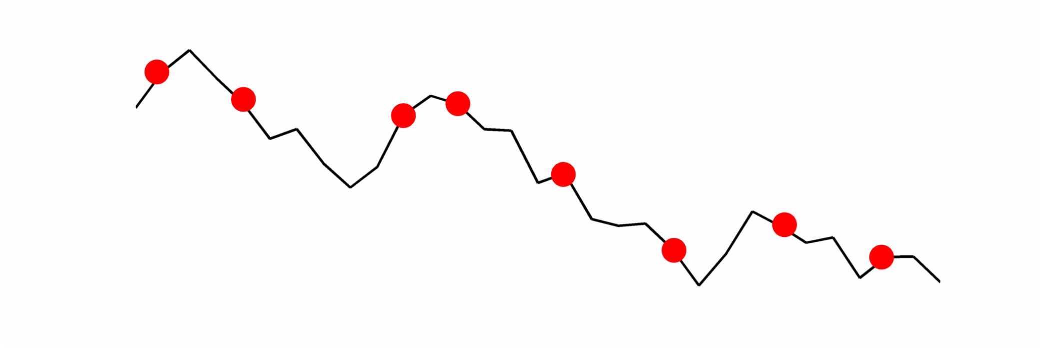
The set G^{ap} is the largest set of peaks in $S_{1.5}$ such that every other peak is skipped. This results in $|G^{ap}| = \text{ceil}(|\mathcal{P}|/2)$. We order the indices of G^{ap} in increasing order with respect to their x-coordinate.

The next result determines the set of vertices visible from G^{ap}.

Proposition 3.1.3 *(Visibility set of G^{ap}): The visibility set of G^{ap} is given by*

$$Q(G^{ap}) = \begin{cases} V & \text{if } |\mathcal{P}| \text{ odd,} \\ \mathcal{J}_{[v_1, p_{|\mathcal{P}|}]} & \text{if } |\mathcal{P}| \text{ even.} \end{cases}$$

Proof 3.1.4 *From Lemma 3.1.1, we deduce that if v_1 and v_2 are two peaks with a single peak v between them, so that $p_+(v_1) = v = p_-(v_2)$, then $Q(v_1 \cup v_2)$ contains $\mathcal{J}_{[p_-(v_1), p_+(v_2)]}$. Thus, $Q(G^{ap})$ is equal to $\mathcal{J}_{[p_-(g_1), p_+(g_{|G^{ap}|})]}$ and the result follows.* $\square$


(a) Agents occupy vertices in G^{ap} on $S_{1.5}$

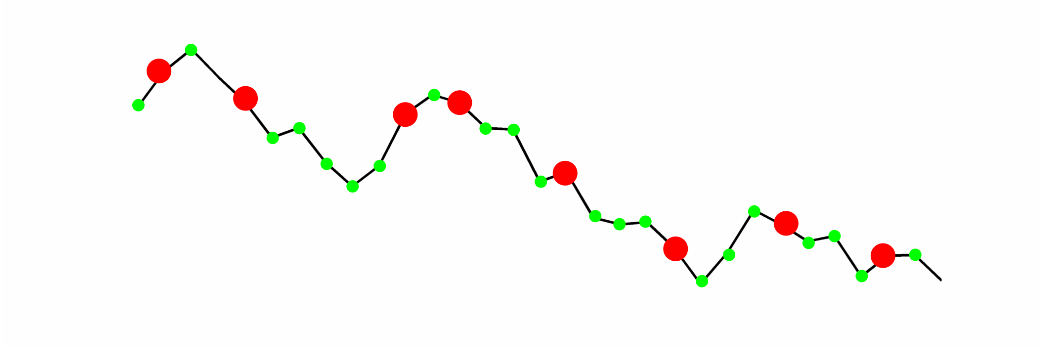

(b) Visibility as a result of occupying G^{ap}

Figure 3.2: Agents occupy vertices in G^{ap} represented by red dots. $Q(G^{ap})$ is represented by green dots.

G^{ap} and $Q(G^{ap})$ is shown in Figure 3.2. Notice in this case that $V_{|V|}$ is not visible to any agents. As a consequence of this result, we identify $G^{ap} \cup \{p_{|\mathcal{P}|}\}$ as a sufficient set of vertices to achieve full visibility.

Theorem 3.1.5 *(Complete visibility): The 1.5D environment $S_{1.5}$ is fully visible from $G = G^{ap} \cup \{p_{|\mathcal{P}|}\}$. Furthermore, $|G| = \text{floor}(|\mathcal{P}|/2)+1$ is sufficient and sometimes necessary to achieve full visibility of $S_{1.5}$.*

Proof 3.1.6 *The fact that $Q(G) = V$ readily follows from Proposition 3.1.3. If $|\mathcal{P}|$ is odd, then $G = G^{ap}$ and therefore $|G| = \text{ceil}(|\mathcal{P}|/2) = \text{floor}((|\mathcal{P}|/2+1)$. If $|\mathcal{P}|$ is even, $|G| = \text{ceil}(|\mathcal{P}|/2) + 1 = \text{floor}((|\mathcal{P}|/2 + 1)$. To show that $|G|$ agents are sometimes necessary, we provide a specific example. Consider an environment where all vertices are peaks. Then, the visibility set of any vertex v is exactly $Q(v) = \mathcal{J}_{[p_-(v),p_+(v)]}$, which implies that any guarding set must contain at least every other vertex in order to achieve full visibility.* $\square$

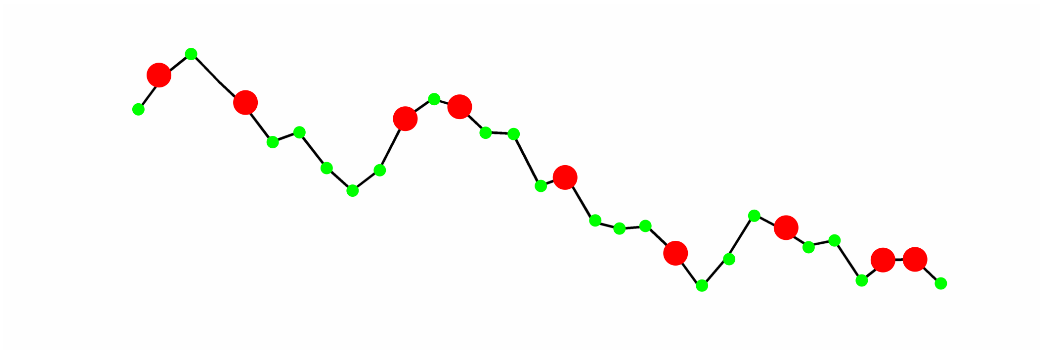


Figure 3.3: A completely guarded $S_{1.5}$ is shown with $G = G^{ap} \cup \{p_{|\mathcal{P}|}\}$. The red dots represent vertices occupied by an agent (v_1 and v_2), and the green dots represent visible vertices in between v_1 and v_2.

3.2 1.5D terrain guarding strategies

Given our analysis in the previous chapter, here we design distributed strategies to deploy $|A| = \text{floor}(|\mathcal{P}|/2) + 1$ agents on $G = G^{ap} \cup \{p_{|\mathcal{P}|}\}$. We begin with an informal description of the algorithm.

> [*Informal description*]: All agents are initially located at vertex, v_0, whose position in $S_{1.5}$ is unknown to them. Agents explore $S_{1.5}$ and incrementally distribute themselves on $G = G^{ap} \cup \{p_{|\mathcal{P}|}\}$. Half of the agents, A_{lft}, go left, while the other half, A_{rght}, goes right (if $|A|$ is odd, we let A_{lft} have one extra agent). Agents in A_{lft} are then given a unique integer ID $\in \{1 \ldots |A_{\text{lft}}|\}$. Agents in A_{rght} are similarly identified. Depending on the location of v_0 within the environment, either A_{lft} or A_{rght} will contain too many agents. Agents keep track of a variable termed "goal". Once an agent detects the edge of $S_{1.5}$ (either v_1 or $v_{|V|}$), it raises its "goal" flag, which signals visible neighboring agents that the other group needs more agents to complete the algorithm. Two strategies are then possible. Let A_- be the group of agents that does not have enough agents, and A_+ be the group that has too many. In both strategies, A_- deploys until they guard as many alternating peaks as they can. Then, in the `1.5D alternate peak strategy with wait`, agents in A_- wait until they receive a "goal" message from A_+ to continue exploring and finally guarding $S_{1.5}$. Instead, in the `1.5D alternate peak strategy w/o wait`, agents in A_- make the assumption that the "goal" flag will eventually come from A_+ and continue deploying towards the boundary of $S_{1.5}$ (creating a void in visibility coverage that will eventually be filled by the agents in A_+).

Algorithm 1 provides a formal description of `1.5D alternate peak strategy with wait` and `1.5D alternate peak strategy w/o wait`. The steps that are only executed under `1.5D alternate peak strategy w/o wait` are marked with the symbol †. All other steps are common to both strategies.

Remark 3.2.1 *(Wait versus no wait):* The strategies differ in how the agents react when they determine that there are not enough agents in their group to reach the boundary of the environment. While the `1.5D alternate peak strategy w/o wait` completes in less time, it requires all agents to start on the same initial condition (otherwise the use of the "continue" flag might be detrimental to algorithm completion). Instead, the `1.5D alternate peak strategy with wait` requires in general more time to complete, but agents can be initialized at multiple locations. •

Algorithm 1 : `1.5D alternate peak strategy`

```
Agent $a$ variables:
bool $goal$=False, $continue$=False
int $direction$=-1 | $a \in A_{\text{lft}}$ or 1 | $a \in A_{\text{rght}}$

While $Q(G)$ is not $V$:
Communicate

    if any visible agents to $a$ have $goal$ is True:
        $a$ sets $goal$ to True
        $a$ sets $direction$ to $direction$ of agent with $goal$ to True

Move

    if any of the following conditions are met:
    • Agent $a$ occupies $v \notin \mathcal{P}$
    • $a \in A_{\text{lft}}$ and $\mathcal{J}_{(v,p_+(v)]}$ is occupied or $a \in A_{\text{rght}}$ and $\mathcal{J}_{[p_-(v),v)}$ is occupied
    • $a$ does not have the greatest ID of all agents that occupy $v$
    • †: $a$ has $goal$ is False and $continue$ is True
        $a$ moves one peak dictated by $direction$
    else:
        $a$ stays at vertex $v$

Detect

    if $v_1$ or $v_{|V|}$ is visible:
        $a$ sets $goal$ to True
        $a$ sets $direction$ away from detected $v_1$ or $v_{|V|}$

†: if the time elapsed is equal to $2A_{-}+a.\text{ID}-2$
    $a$ sets $continue$ to True
```

Remark 3.2.2 *(Ordering of agents):* Agents with lower unique IDs occupy a peak only if no other agent with lower ID occupies the same peak. As the algorithm executes, the agents are then ordered within their respective groups of A_- and A_+ in decreasing order of ID from v_0 in the direction they are initialized. This allows the agents to determine when the "goal" flag should have arrived by (agents in A_+ with lower ID receive the "goal" flag before agents with greater ID). Due to the speed at which the "goal" flag propagates in A_+, agents in A_- rationalize that they are not in A_+ if they do not receive the "goal" flag in $2A_-+$ ID -2 time steps. •

Figure 3.4 shows an example of agents being deployed on $S_{1.5}$ using the `1.5D alternate peak strategy with wait`. At time step: 4, A_+ reaches the leftmost boundary and raises the "goal" flag. At time step: 7, A_- runs out of agents and begins to wait for the "goal" flag. By time step: 15, the network has completely deployed achieving full visibility of the environment.

3.3 1.5D terrain guarding strategy time analysis

In this chapter we characterize the number of time steps required by the proposed strategies for completion. We recall that an agent can move between adjacent peaks in one time step. For the following analysis, let i be the index of v_0 in $\mathcal{P}$ and define

$$i^* = \begin{cases} i & \text{if } i \leq |\mathcal{P}|/2, \\ |\mathcal{P}| - i + 1 & \text{if } i > |\mathcal{P}|/2. \end{cases} \tag{3.1}$$

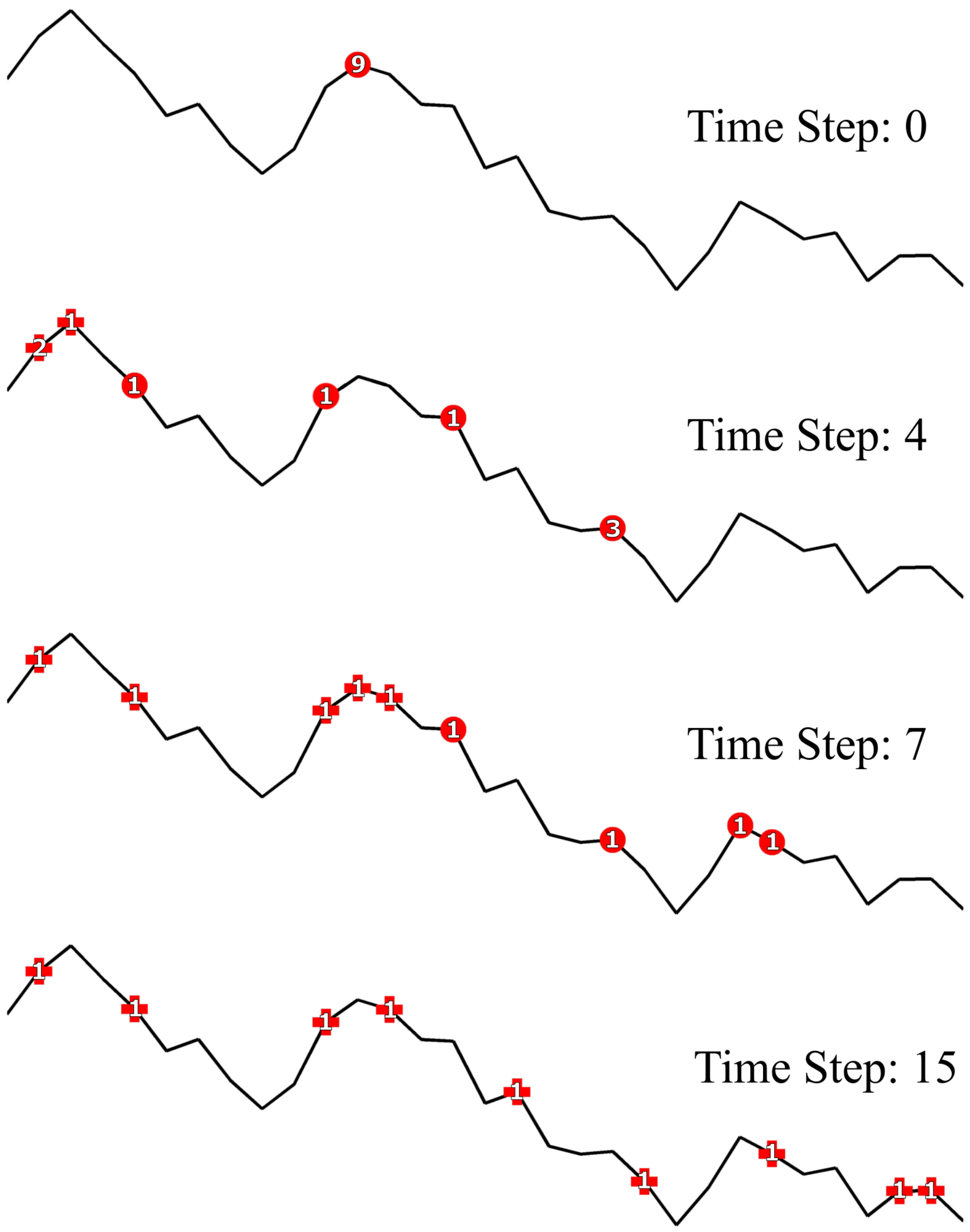


Figure 3.4: Execution of `1.5D alternate peak strategy with wait` on a 1.5D environment with 16 peaks. From Theorem 3.1.5, $|A| = 9$ agents are sufficient to achieve full visibility. All agents begin at the same initial location, v_0, of index 6 with respect to $\mathcal{P}$, and split into two groups. The location of agents with the "goal" flag not raised are shown by a red dot where the number states the number of agents on that vertex. Agents with a raised "goal" flag are denoted with a red cross.

The following three sets cover all possibilities for the location of the initial vertex v_0,

$$\mathcal{A} = \{v_0 \,|\, \text{if } |A| \text{ even}, i \leq |\mathcal{P}|/6 + 1 \text{ or } i \geq 5|\mathcal{P}|/6 - 1$$

$$\text{and if } |A| \text{ odd}, i \leq |\mathcal{P}|/6 \text{ or } i \geq 5|\mathcal{P}|/6\},$$

$$\mathcal{B} = \{v_0 \,|\, \text{if } |A| \text{ even}, |\mathcal{P}|/6 + 1 < i < 5|\mathcal{P}|/6 - 1$$

$$\text{and if } |A| \text{ odd}, |\mathcal{P}|/6 < i < 5|\mathcal{P}|/6\},$$

and region $\mathcal{C}$, which only exists if $|\mathcal{P}|$ is odd and corresponds to $i = \frac{|\mathcal{P}|+1}{2}$. Figure 3.5 illustrates these three cases. We are ready to characterize the time complexity of the

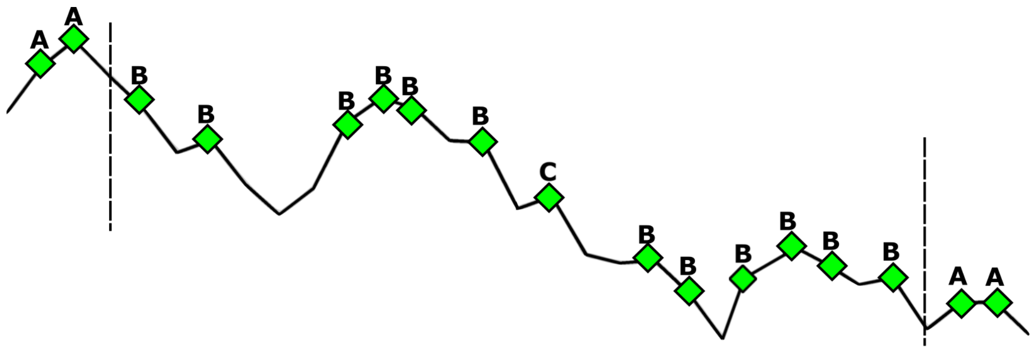


Figure 3.5: Illustration of cases $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ for the locations of the common initial condition of the agents. The 1.5D environment $S_{1.5}$ has $|\mathcal{P}| = 17$.

strategy with wait.

Theorem 3.3.1 *(`1.5D alternate peak strategy with wait` completion time): The number of time steps required by the* ***`1.5D alternate peak strategy with wait`*** *to complete is*

$$T = \begin{cases} |\mathcal{P}| - i^* & v_0 \in \mathcal{A}, \\ |\mathcal{P}| + \frac{i^*}{2} - |A_-| - \frac{3}{2} & v_0 \in \mathcal{B}, \\ \frac{3(|\mathcal{P}|-1)}{4} & v_0 \in \mathcal{C}. \end{cases} \tag{3.2}$$

Proof 3.3.2 *For simplicity of exposition, we only consider the case when $i \leq (|\mathcal{P}|+1)/2$ (the case when $i > \frac{|\mathcal{P}|}{2}$ is analogous). Consequently, $i^* = i$, $A_{lft} = A_+$, and $A_{rght} = A_-$. We first consider the scenario when both groups of agents reach the boundary of the environment at the same time. Note that this is only possible if $v_0 \in \mathcal{C}$.*

***Case** $\mathcal{C}$: If $|\mathcal{P}|$ is odd and v_0 is the peak with index $\frac{|\mathcal{P}|+1}{2}$, the agents split up perfectly since there are the same number of peaks to the left and right. The agents reach the boundaries and send the "goal" message at the same time. The algorithm completes when the goal messages meet at $\frac{|\mathcal{P}|+1}{2}$. The time to completion is then the sum of the time to reach the boundaries, and the time that it takes for the flags to reach the $\frac{|\mathcal{P}|+1}{2}$, which is*

$$\frac{|\mathcal{P}|+1}{2} - 1 + \frac{\frac{|\mathcal{P}|+1}{2} - 1}{2} = \frac{3(|\mathcal{P}|-1)}{4}.$$

Next, we consider the scenario when both groups of agents do not reach the boundary of the environment at the same time. In this scenario, it is A_+ which reaches the boundary first. Agents move one peak at a time, distributing themselves on every other peak. Because of this, note that A_- runs out of agents after exactly $2|A_-|$ time steps. On the other hand, it takes exactly $i^* - 1$ time steps for agents in A_+ to reach the boundary of $S_{1.5}$ and raise the "goal" flag. At this time, the rightmost agents in A_- are located at peak $i^* + (i^* - 1) = 2i^* - 1$. Once the "goal" flag is raised, since agents can communicate with agents at adjacent peaks, the speed at which the "goal" flag is communicated is effectively two peaks per time step.

Two things might happen depending on whether or not the "goal" flag reaches the rightmost agents in A_- before this group runs out of agents. Let t denote the *number of time steps elapsed since A_+ first raised the "goal" flag. After* t *time steps, the goal flag is at* $1+2t$*. If* A_- *does not run of agents, its rightmost agents are at* $2i^*-1+t$*. Therefore, we are looking for the solution to* $1+2t = 2i^*-1+t$*, which is*

$$t = 2i^* - 2.$$

The total elapsed time since the beginning is then $i^*-1+(2i^*-2) = 3i^*-3$*. This time must be less than or equal to than the time it takes* A_- *to run out of agents, i.e.,*

$$i^* \leq \frac{2}{3}|A_-| + 1. \tag{3.3}$$

Case $\mathcal{A}$*: One can see that equation* (3.3) *is satisfied if and only if* $v_0 \in \mathcal{A}$*. Because the "goal" flag reaches the rightmost agent in* A_- *before* A_- *runs out of agents,* A_- *moves at one time step towards the rightmost boundary through the entirety of the strategy. Once* A_- *reaches the boundary, the agents will have distributed themselves on* G*. Therefore, if* $v_0 \in \mathcal{A}$*, we deduce that the number of time steps required for completion is* $T = |\mathcal{P}| - i^*$*.*

Case $\mathcal{B}$*: If instead,* $v_0 \in \mathcal{B}$*, this means that equation* (3.3) *is not satisfied, i.e.,* A_- *runs out of agents before the "goal" flag reaches its rightmost agents. After the "goal" flag is raised, agents in* A_+ *move at 1 peak per time step and occupy their half of* G *by the time the "goal" flag reaches the rightmost boundary, since no agent has to travel more than* $|\mathcal{P}|/2$ *peaks. Since agents in* A_- *previously occupy alternating peaks, they all must travel the same number of peaks to reach their final configuration. The rightmost agent in* A_- *receives the "goal" flag last and is the last agent to occupy its peak in* G*. Therefore, we need to compute the time it takes for* A_- *to receive the* *message and the leftover time needed for the rightmost agent in A_- to move to the boundary of $S_{1.5}$. A_- runs out of agents at vertex $d = i^* + 2|A_-|$. With the notation used above, the time required for the "goal" flag to reach this vertex is the solution to $1 + 2t = d$, i.e., $t = (d-1)/2$. Once the A_- has received the message it takes*

$$|\mathcal{P}| - d,$$

steps to reach the boundary. Therefore, the total number of time steps is

$$T = i^* - 1 + (d-1)/2 + |\mathcal{P}| - d = |\mathcal{P}| + \frac{i^*}{2} - |A_-| - \frac{3}{2}. \qquad \square$$

From Theorem 3.3.1, one can see that, in region $\mathcal{B}$, the time complexity monotonically increases as the initial location moves from the left boundary of this region (at $|\mathcal{P}|/6+1$ or $|\mathcal{P}|/6$, depending on whether $|A|$ is even or not), to the peak closest to $\frac{|\mathcal{P}|}{2}$, Next, we determine the completion time of the `1.5D alternate peak strategy w/o wait`.

Theorem 3.3.3 *(`1.5D alternate peak strategy w/o wait` completion time): The number of time steps required for the `1.5D alternate peak strategy w/o wait` to complete is*

$$T = \begin{cases} |\mathcal{P}| - i^* & v_0 \in \mathcal{A}, \\ \frac{7|\mathcal{P}|}{8} - \frac{i^*}{4} & v_0 \in \mathcal{B}, \\ \frac{3(|\mathcal{P}|-1)}{4} & v_0 \in \mathcal{C}. \end{cases} \tag{3.4}$$

Proof 3.3.4 *With respect to the proof of Theorem 3.3.1, the scenarios when the initial* *condition belongs to $\mathcal{A}$ and $\mathcal{C}$ are the same. In case $\mathcal{A}$, the "goal" flag is communicated completely before either group runs out of agents. Similarly, in case $\mathcal{C}$ the algorithm completes before the "continue" flag is raised.*

***Case** $\mathcal{B}$: In this case, agents in A_- determine as soon as possible that they are in A_- and raise their "continue" flag. Agent, a, in A_- raises its "continue" flag at $2A_-+ID\ -2$ time steps, where ID is the unique identification of the agent. Since the agents deploy in order of decreasing ID, this is the amount of time the "goal" flag should have reached a if a belongs A_+. a then moves at one peak per turn in its initial direction until it receives the "goal" flag. When furthest agent in A_- reaches the boundary it raises the "goal" flag. This means that if v_0 is in region $\mathcal{B}$, two "goal" flags may be active at the same time and propagate from the boundaries to the center. The algorithm terminates when agents no longer move, which occurs where the two "goal" flags meet. We use this fact to determine the time of execution for `1.5D alternate peak strategy w/o wait` in region $\mathcal{B}$. For this analysis, we examine the "goal" flag from A_-. The rightmost agent in A_- reaches the boundary and raise the "goal" flag in $t_1 = |P| - i^*$ since the `1.5D alternate peak strategy w/o wait` allows agents to move at one peak per time step to the boundary when "continue" is raised. The rightmost "goal" flag travels at two peaks per time step and meets the leftmost "goal" flag in $t_2 = \frac{d}{2}$ time steps, where d is the distance from the rightmost boundary to where the "goal" flags meet. To determine d, we interpolate between two boundary initial conditions of region $\mathcal{B}$. For $v_0 = \frac{|P|}{2}$, A_- and A_+ reach their respective boundaries at or around the same time. The goal flags meet at the center of $S_{1.5}$, and $d = \frac{|P|}{2}$. For $v_0 = \frac{|P|}{6}$, the goal flag from A_+ reaches the rightmost agent in A_- just as the "continue" flag is raised and meets with rightmost "goal" flag* *at* $d = 0$. *We interpolate for:*

$$d = \frac{3}{4}(i^* - \frac{|\mathcal{P}|}{6})$$

which allows us to determine the time of execution:

$$T = t_1 + t_2 = |\mathcal{P}| - i^* + \frac{3}{4}(i^* - \frac{|\mathcal{P}|}{6}) = \frac{7|\mathcal{P}|}{8} + \frac{i^*}{4} \qquad \square$$

Chapter 3, in part, has been submitted for publication of the material as it may appear in the Dynamic Systems and Control Conference, 2016, Cortés, Jorge. The thesis author was the primary investigator and author of this paper.

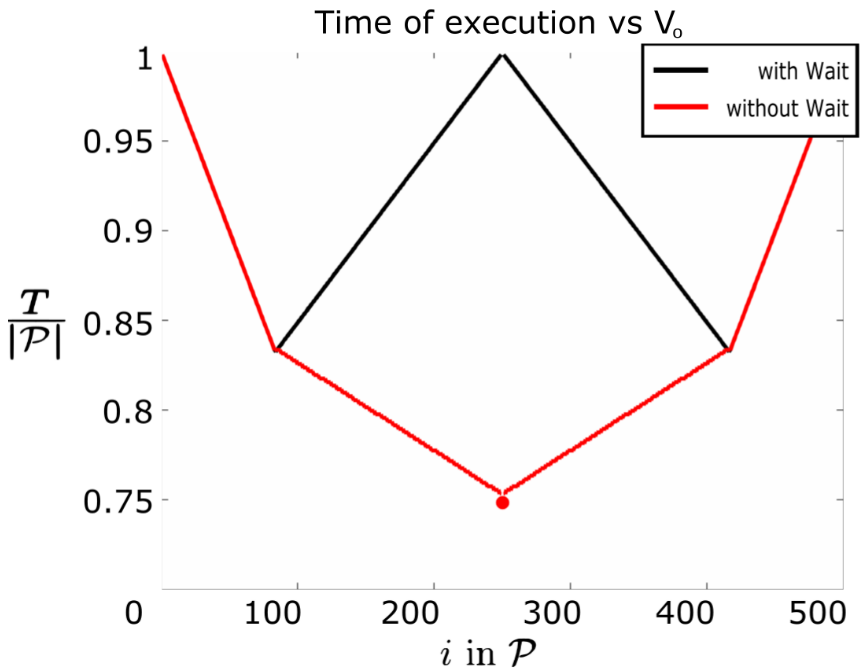


Figure 3.6: The time of execution vs. v_0 is shown for $S_{1.5}$ with 500 peaks. On the y-axis we list the T, the time of execution, over $|\mathcal{P}|$, the number of peaks. The x-axis describes the initial starting location v_0.

Chapter 4

2.5D terrain guarding

This chapter studies the distributed deployment problem over 2.5D polyhedral terrains. We introduce the concept of (non-)redundant vertex of a guarding set and characterize a sufficient and sometimes necessary number of vertices of guarding sets without redundant vertices. We build on this result to design a distributed strategy to efficiently place the robotic agents and achieve full visibility.

4.1 Guarding set via non-redundant vertices

$S_{2.5}$ contains many sets of vertices, G, such that $Q(G) = V$. We begin by defining a reducible set of vertices that are analogous to valleys in $S_{1.5}$. In this chapter we determine $S_{2.5}^{**}$, a planar graph determined by contracting reducible sets in $S_{2.5}^{*}$. Next we define a redundant vertex and quantify how many non-redundant vertices can exist in $S_{2.5}^{**}$.

Consider a set of vertices, $\mathcal{R}^*$, such that all vertices within the convex hull of $\mathcal{R}^*$ are visible to each other. Refer to $\mathcal{R}_{hull}$ as the set of vertices that contribute to the convex hull of $\mathcal{R}^*$, and $\mathcal{R}$ as the set of all other vertices, $\mathcal{R} = \mathcal{R}^* \backslash \mathcal{R}_{hull}$. We call $\mathcal{R}$ a *reducible set*. We create a new planar graph, $S_{2.5}^{**}$, which is a modification of $S_{2.5}^{*}$, where every $\mathcal{R}$ in $S_{2.5}^{*}$ contracts into a vertex as shown in Figure 4.1. The vertices that remain in $S_{2.5}^{**}$ are then V^{**}.

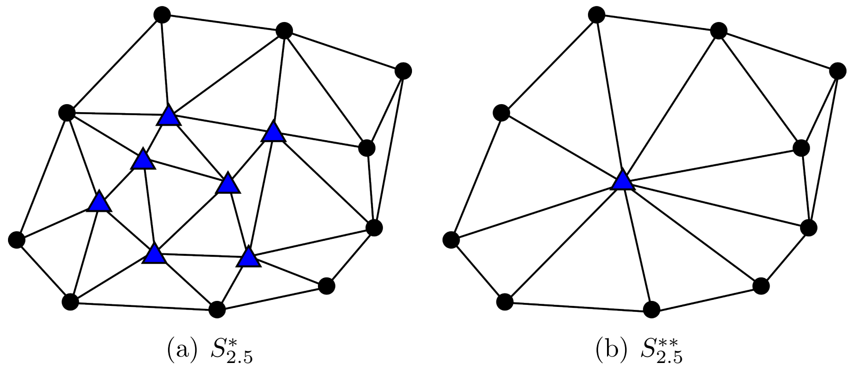

(a) $S_{2.5}^{*}$ (b) $S_{2.5}^{**}$

Figure 4.1: Here is an example of a reducible set in $S_{2.5}^{*}$ being contracted. Assume that the blue triangles represent vertices in $\mathcal{R}$. $|\mathcal{R}| = 7$ is reduced to $|\mathcal{R}| = 1$ after contraction.

We now define a redundant vertex in $S_{2.5}$.

Definition 4.1.1 *(Redundant vertex): Consider a vertex, v, that is occupied in guarding set G. Define Δ_v as the set of triangles, t, in contact with v as determined by vertices and edges in $S_{2.5}^{**}$. v is a redundant vertex if there is another occupied vertex that is able to see* t *for all* $\mathrm{t} \in \Delta_v$. *Conversely v is non-redundant, if and only if there exists* $\mathrm{t} \in \Delta_v$ *such that* t *is not visible to any other occupied vertex. Furthermore, if v is a redundant vertex with respect to some guarding set, G, then $Q(G) = Q(G \backslash v)$.*

Let Γ denote the set of all non-redundant vertices and let *non-redundant pairs*, p, be two vertices, v_1 and v_2, that define v to be non-redundant. v_1 and v_2 are connected with each other and v to form a triangle uniquely visible to v. Both v_1 and v_2 must be unoccupied to satisfy that v is a non-redundant vertex. In general, if v is non-redundant, it can have any number of p that belong to set $\mathrm{P}(v)$. Vertices that are unoccupied, but do not have any unoccupied neighbors belong to U. A pair of vertices form an edge that belongs to, at most, two triangles in $S^{**}_{2.5}$. By manipulating Γ such that pairs are shared among two non-redundant agents, we are able to maximize the Γ.

Theorem 4.1.2 *(Upper bound of $|\Gamma|$): The maximum number of non-redundant vertices is* $|\Gamma| = |2V^{**}|/3$.

Proof 4.1.3 *Figure 4.2 shows the worst-case scenario that yields* $|\Gamma| = 2|V^{**}|/3$*, where every vertex in* Γ *only contains one pair that defines them as non-redundant. Every pair is shared by two non-redundant vertices and* $\mathrm{U} = \emptyset$. $\square$

For the following deployment strategy, we guard non-redundant vertices. We conclude that $|A| = 2|V^{**}|/3$ is sufficient and sometimes necessary as a result of Theorem 4.1.2.

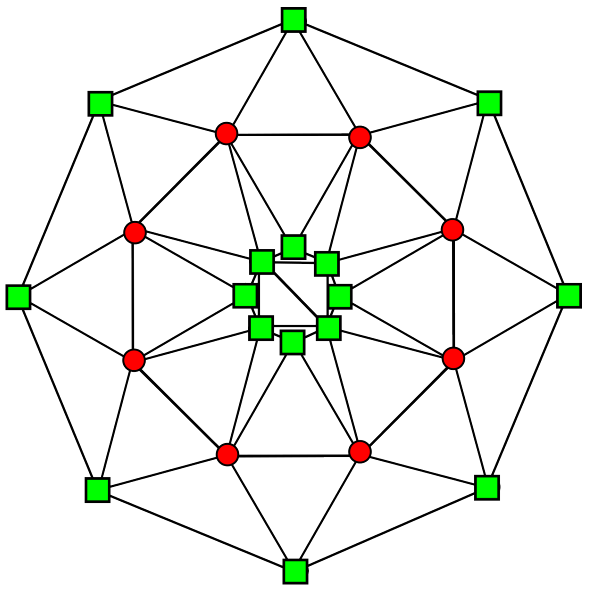


Figure 4.2: The configuration that contains the maximum number of non-redundant vertices in $S^{**}_{2.5}$. Here, the green squares are non-redundant vertices, and the red circles are redundant vertices.

4.2 2.5D terrain guarding strategy

Given $|A| = 2|V^{**}|/3$ we determine an algorithm for distributed coverage of $S_{2.5}$. We design an algorithm that distributes agents on G, determined by a general coloring of $S_{2.5}^{**}$. Our strategy for coordinated exploration and guarding entails maintaining strong connectivity of the communication between agents and relays at any given time of deployment. We allow the agents to place relays, $r \in \mathrm{R}$, which provide communication between neighboring vertices. The agents start on a single vertex and detect vertices in $S_{2.5}$ as they explore. Let $\mathcal{U} = V \backslash Q(G)$ be the set of vertices that the agents have not detected yet. Let $\mathcal{K}$ be the set of vertices that are not in $\mathcal{U}$ and have no neighbors in $\mathcal{U}$. Finally let $\mathcal{D}$ be the set of vertices that are not in $\mathcal{U}$ but have neighbors in $\mathcal{U}$. Agents assign colors to the vertices as they are detected ($v \notin \mathcal{U}$). We refer to the colors that the agents assign to vertices as $\mathrm{COL} = \{\mathit{1}, \mathit{2}, G_c\}$, where G_c is a label for a color in $\{\mathit{3}, \mathit{4}, \mathit{5}, \mathit{6}\}$ that is instantiated during the execution of the algorithm and denotes vertices that agents plan to occupy or place a relay ($Q(G) = Q(G_c)$).

Lemma 4.2.1 *(Three-coloring of a triangle): Every triangular face in $S_{2.5}^{**}$ contains a vertex labeled a color in G_c after coloring.*

Proof 4.2.2 *Vertices in a triangle are all connected. Therefore a triangle must be colored with three unique colors. Since we color the vertices, $\{\mathit{1}, \mathit{2}, G_c\}$, at least one vertex must be labeled a color in G_c.* $\square$

Occupying all vertices that are not $\{\mathit{1}, \mathit{2}\}$ guarantees complete visibility, since every face on $S_{2.5}$ is a triangle. We proceed to define `2.5D non-redundant peak strategy`:

[*Informal description*]: $|A|$ agents start on v_0 that is assigned color G_c. All neighboring vertices are visible and are put into a set, D. D contains vertices that have been detected, but not yet colored. Once all vertices in D are colored, D is set to $\emptyset$. While exploring, the agents color vertices in D and keep track of a graph, $\mathcal{C}$, that contains vertices and edges they detect, as well as the colors they assign. Agents color a vertex with either *1* or *2* if possible. If that is not possible, the agents non-uniquely label the vertex G_c. The agents simultaneously explore $S_{2.5}$ and create a tree $\mathcal{T}$ with v_0 as the root. Agents look one at a time for an unoccupied (by relay or agent) vertex v in $\mathcal{D}$ of color G_c if it exists. If it does not exist, the agents find a vertex $\in \mathcal{D}$ with color in $\{1, 2\}$. The agent that finds v, moves to it. The color of v is changed from either *1* or *2* to G_c. If a child agent of a with respect to $\mathcal{T}$ moves, then a moves to the vertex that the child agent last occupied. Agents then use Algorithm 3 to refine the guarding set so that $|A|$ is sufficient. We consider the coloring and refining guarding set processes to take negligible time with respect to the exploration routine. Finally the $S_{2.5}$ Deployment algorithm repeats until agents occupy G such that $Q(G) = V$.

Remark 4.2.3 *(Exploration):* As a result of moving to the exploration process, vertices in $S^{*}_{2.5}$ are discovered. These vertices are visible and now belong to $Q(G)$ through completion of the `2.5D non-redundant peak strategy`. When new vertices are discovered, agents re-evaluate $S^{**}_{2.5}$ and D. •

As a result of the `2.5D non-redundant peak strategy` we define $S^{**}_{2.5G}$ as the subgraph that is the result of removing v that are not labeled G_c and their corresponding edges from $S^{**}_{2.5}$. $S^{**}_{2.5G}$ contains only vertices that are occupied by an agent or relay. We continue to uniquely color $S^{**}_{2.5G}$ with colors in $G_c = \{\mathit{3}, \mathit{4}, \mathit{5}, \mathit{6}\}$. Colors in $S^{**}_{2.5G}$ are applied to $S^{**}_{2.5}$ which results in unique labeling of $S^{**}_{2.5}$. We determine the number of colors, |COL| sufficient and sometimes necessary for labeling $S^{**}_{2.5}$. The upper bound of |COL| follows:

Theorem 4.2.4 *The number of colors sufficient and sometimes necessary for color-*

Algorithm 2 : `2.5D non-redundant peak strategy`

```
Agent $a$ variables:
bool explored=False
int ID

While $Q(G)$ is not $V$:
Coloring

  if explored is True:
      for all $v \in$ D that neighbor $a$:
        if $v$ has no neighbors with color 1:
            $a$ colors $v$ to 1
        else if $v$ has no neighbors with color 2:
            $a$ colors $v$ to 2
        else:
            $a$ colors $v$ to $G_c$
  $a$ sets explored to False
  $a$ broadcasts and updates colors in D
  $a$ sets D to $\emptyset$

Explore

  if an unoccupied vertex $v \in \mathcal{D}$ of color $G_c$ exists:
     if $a$ has the lowest ID that neighbors $v$
         $a$ moves to $v$
         $a$ broadcasts and updates tree
         $a$ sets explored to True
  else if $a$ neighbors $v \in \mathcal{D}$ of color 1 or 2:
      if no agents with lower ID have moved this time step
         $a$ moves to $v$
         $a$ broadcasts and updates tree
         $a$ sets explored True
  if a child agent, $a_c$, of $a$ moved:
      $a$ moves to last occupied vertex of $a_c$

Refine guarding set

  execute Algorithm 3
```

ing $S^{**}_{2.5}$ is given by

$$|\mathrm{COL}| = \begin{cases} 4, & \textit{if } S^{**}_{2.5G} \textit{ contains no cycles} \\ 5, & \textit{else if } S^{**}_{2.5G} \textit{ contains no three-cycles,} \\ 6, & \textit{else.} \end{cases}$$

Proof 4.2.5 *It is straightforwarded that a graph that contains no cycles can be 2-colored. with no cycles present, only $\{3, 4\}$ are necessary to color $S^{**}_{2.5G}$, which results in a 4-coloring of $S^{**}_{2.5}$. Proof that a planar graph with no three-cycles (triangles) is always 3-colorable is shown in [14], resulting in a 5-coloring of $S^{**}_{2.5}$. It is known that any planar graph is 4-colorable, and since $S^{**}_{2.5G}$ is necessarily planar, it is always 4-colorable.*

The resulting $S^{**}_{2.5G}$ is shown in Figure 4.5. In the case presented, $S^{**}_{2.5G}$ contains 3-cycles so we are only able gaurantee an upper bound of $|\mathrm{COL}| = 6$, ($|\mathrm{COL}| = 4$ for $S^{**}_{2.5G}$), however we easily color $S^{**}_{2.5G}$ with 3 colors, and $S^{**}_{2.5}$ with 5 colors as shown in Figure 4.6. We continue to prove termination of `2.5D non-redundant peak strategy` with $Q(G) = V$.

Lemma 4.2.6 *(Completion of the `2.5D non-redundant peak strategy`): The exploration process results in complete exploration of $S_{2.5}$.*

Proof 4.2.7 *The agents are able to determine if vertices belong to $\mathcal{K}$ or $\mathcal{D}$ since they can determine if all planes surrounding a vertex are visible or not. $\mathcal{U} \neq \emptyset$ and $\mathcal{D} = \emptyset$ cannot both be true therefore, until $\mathcal{U} = \emptyset$, the agents are always able to move towards a vertex in $\mathcal{D}$. Thus, $|Q|$ monotonically increases by at least 1 every turn until $Q(G) = V$.* $\square$

We proceed to define Algorithm 3:

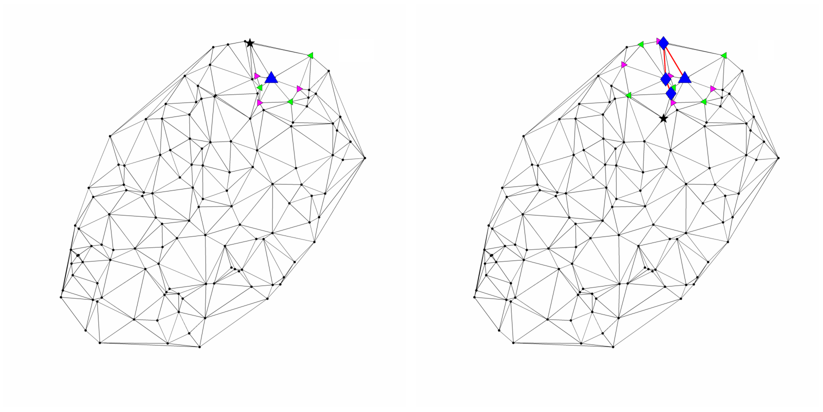
> [*Informal description*]: Agent, a, independently determines if it is a redundant agent by checking if another active agent shares a triangle in Δ_v. If a is redundant, it broadcasts that it is a redundant and counts the number of neighboring redundant neighbors, n. Then a broadcasts n. If a is redundant and has the lowest n of all agents, then a places a relay on its vertex. If there are multiple agents that have the lowest number of neighboring redundant neighbors, then the agent with the lowest index in A executes this action.

Algorithm 3 : `Redundant agent removal`

```
Agent $a$ variables:
bool $redundancy$ = True
int $n = 0$

While an agent occupies a redundant vertex:
Calculate

  if there exists $t \in \Delta_v$ that is exlusive to $a$:
    $a$ sets $redundancy$ to False

Communicate

  if $redundancy$ is True:
    $a$ communicates $redundancy$ to neighbors
    for each neighbor with $redundancy$ equal to True
      $n = n+1$

Place relay

  if $redundancy$ is True and $n$ least of all redundant agents
    $a$ places a relay at $v$
    $a$ no longer occupies $v$
```

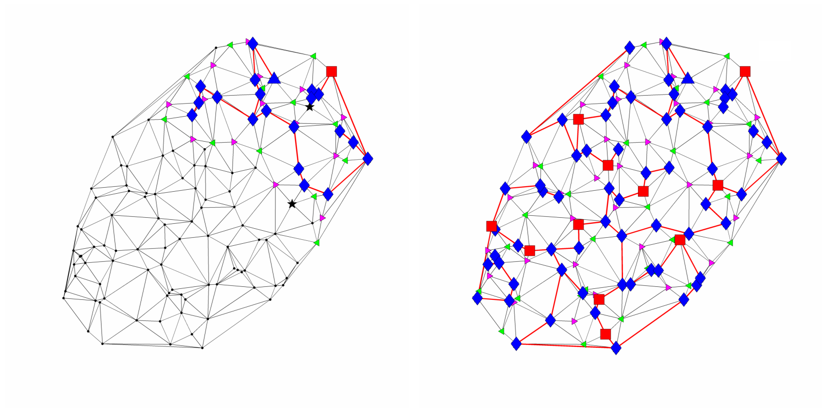
Remark 4.2.8 *(`Redundant agent removal`):* By construction, this process guarantees $|Q|$ never decreases. It is necessary that Algorithm 3 be recursive because the set of redundant agents changes every time a redundant agent is removed. •

Finally, we determine that `2.5D non-redundant peak strategy` will complete in under $|V^{**}| - 1$ time steps.



(a) Initialization of the `2.5D non-redundant peak strategy`

(b) Exploration and coloring after one time step



(c) Exploration and coloring after 4 time steps

(d) Fully guarded $S_{2.5}$

Figure 4.3: Execution of `2.5D non-redundant peak strategy` on $S^{**}_{2.5}$ with 135 vertices. In (a) the agents start on arbitrary vertex, v_0, represented by blue triangles. The vertices that are visible to the agents are denoted with black dots. The agents label the newly detected vertices surrounding v_0 with colors priority: *1* = green triangles, *2* = pink triangles, and G_c = black stars. In (b), one agent moves to the unoccupied vertex with color G_c, and colors the newly discovered vertices. The vertices that the agents actively guard are represented with blue diamonds and the shift of the robots is represented by a black arrow. In (c), four time steps have passed as the agents continue deployment. At this point, one of them detects that it occupies a redundant vertex, v. In order to resolve this, the agent places a relay, represented by red squares, at v to maintain communication to the agents and to notify other agents not to occupy v. Finally in (c), after 74 time steps, the `2.5D non-redundant peak strategy` completes with 63 active agents and 11 relays.

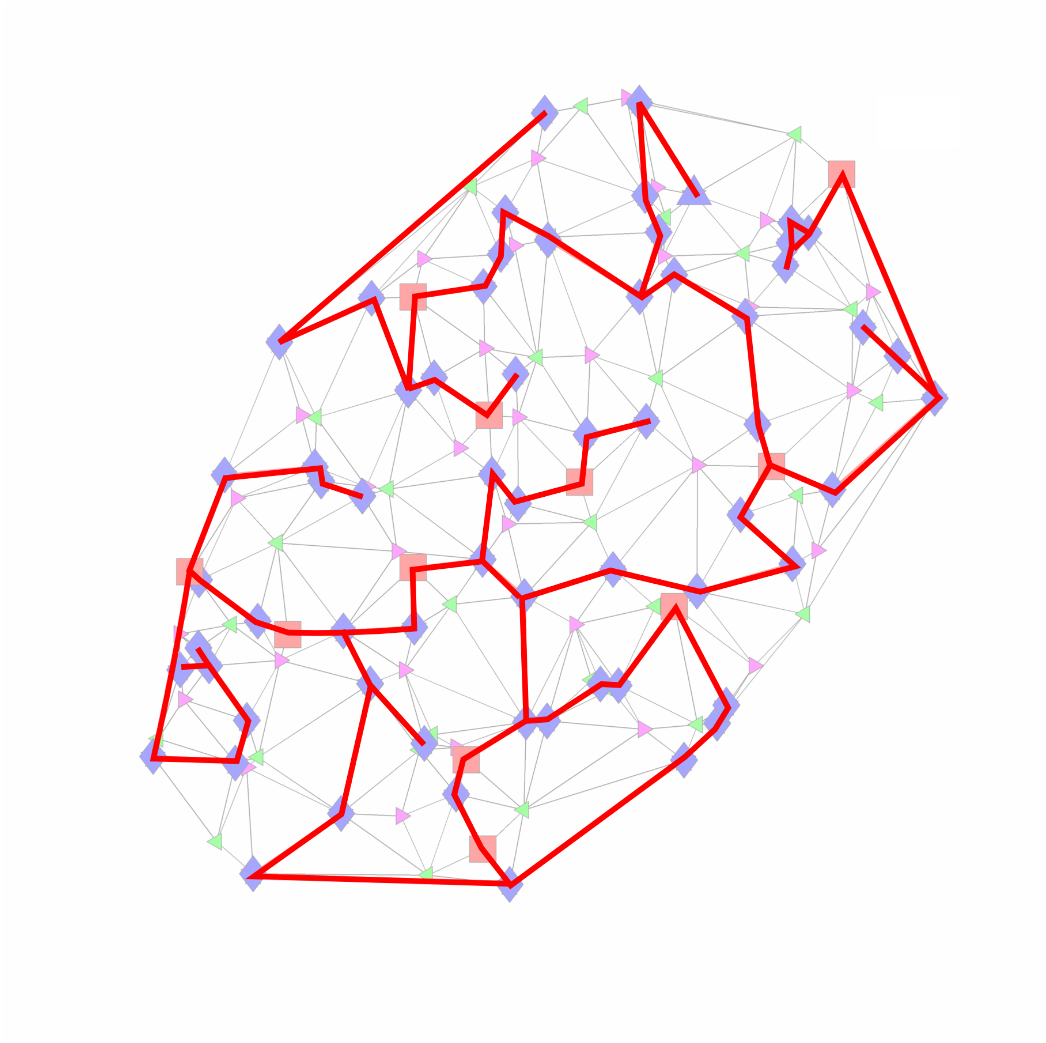


Figure 4.4: The communication network is shown with emphasized red lines. The network created by `2.5D non-redundant peak strategy` is strongly connection.

4.3 2.5D terrain guarding strategy time analysis

We characterize the time steps for execution of `2.5D non-redundant peak strategy`. We provide a coloring scheme such that we label vertices with priority $\{1, 2, G_c\}$. In general, the algorithm will complete for any coloring scheme, not just the one that we provide. Therefore, the upper and lower bounds on the completion time is determined to cover any imaginable coloring scheme that is applied to the strategy, as long as $\{1, 2\}$ are uniquely labeled. The number of time steps is ulti-

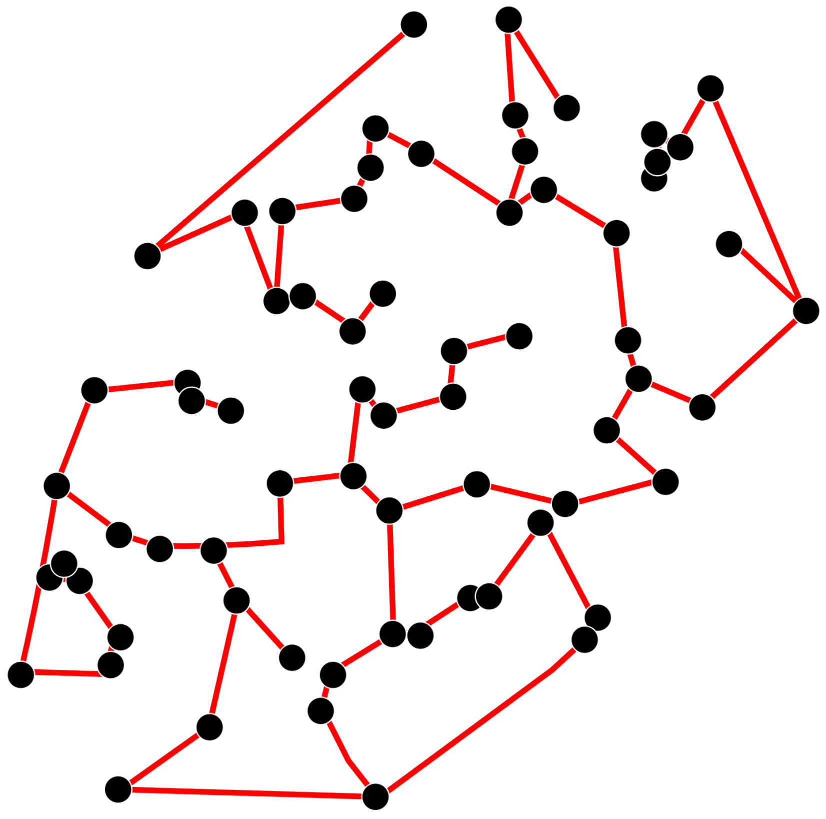


Figure 4.5: $S^{**}_{2.5G}$ is shown.

mately determined by the color scheme(including which vertices to convert from *1* or *2* to G_c if no unnocupied G_c are available during the exploration phase), which dictates efficacy of exploration.

Theorem 4.3.1 *(The `2.5D non-redundant peak strategy` completion time): The $S_{2.5}$ deployment strategy takes at most $|V^{**}| - 3$ time steps to complete.*

Proof 4.3.2 *The strategy completes when $Q(G) = V$. From Lemma 4.2.6, $|Q|$ is increased by at least 1 every turn. Furthermore, the coloring and refining of the guarding set processes do not reduce $|Q|$. Therefore, the completion time is dictated*

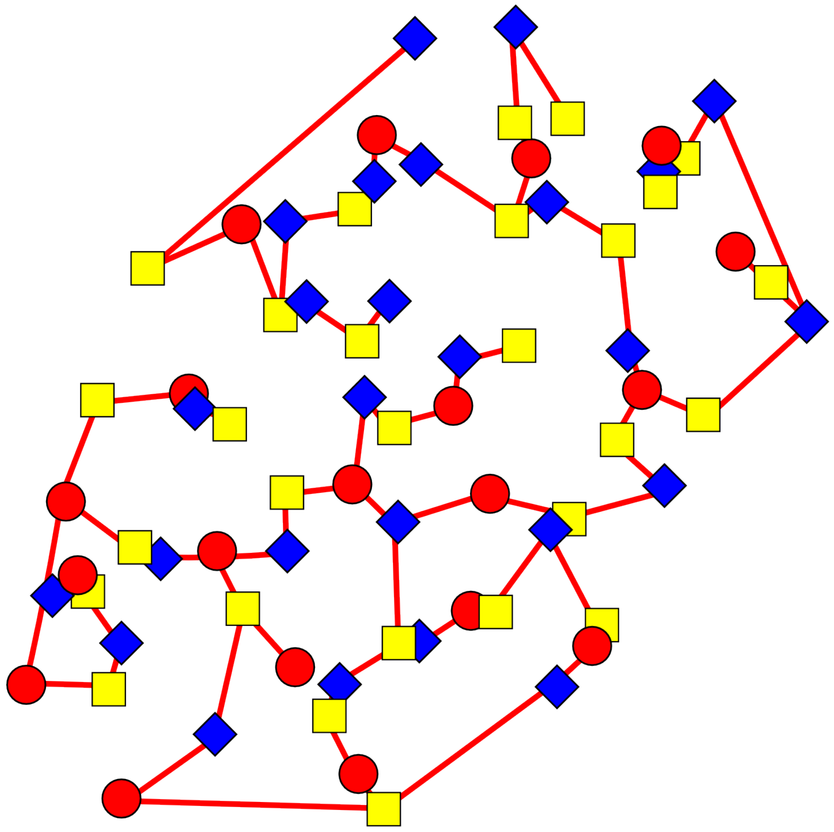
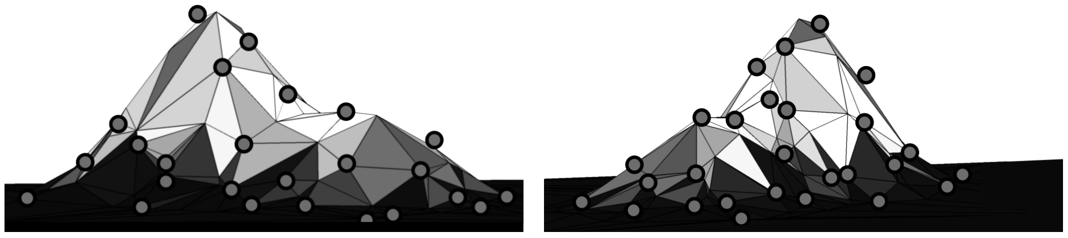


Figure 4.6: $S^{**}_{2.5G}$ is shown with colors. Even though there are three-cycles, and we can only gaurantee that $S^{**}_{2.5G}$ is no more than 4-colorable, we are able to color the resulting graph with only 3 colors.



(a) Fully guarded $S_{2.5}$ $(0°)$ (b) Fully guarded $S_{2.5}$ $(120°)$

Figure 4.7: Results of `2.5D non-redundant peak strategy` in $S_{2.5}$ are shown.

by how quickly the agents explore $S^{**}_{2.5}$. Since Q includes v_0, and at least two other vertices (since every vertex has at least two neighbors in the polyhedral terrain), the strategy takes no more than $|V^{**}| - 3$ time steps to complete. $\square$

Although it is possible for the algorithm to take up to $|V^{**}| - 3$ time steps, it is highly unlikely and can be avoided through smarter coloring schemes. We provide an example the worst case scenario as shown in Figure 4.8. In this execution we use a color scheme that prioritizes labeling by $\{1, 2, G_c\}$. During the exploration phase, there are no unnocupied vertices with color G_c, therefore the agent has to label a vertex with color *1* or *2* to G_c and proceed to move to that vertex.

We determine the lower bounds on completion time of the `2.5D non-redundant peak strategy`.

Theorem 4.3.3 *(The `2.5D non-redundant peak strategy` completion time): The $S_{2.5}$ deployment strategy takes at least $\frac{\#f-2}{d_{out}-2} - 1$ time steps to complete, where $\#f$ is the number of faces and d_{out} is the max out-degree of vertices in $S^{**}_{2.5}$.*

Proof 4.3.4 *To prove Theorem 4.3.3, we start by defining a face-spanning subgraph as a tree in $\mathcal{G}$ that contain atleast one vertex on the boundary of every face in $\mathcal{G}$. The `2.5D non-redundant peak strategy` creates $S^{**}_{2.5G}$ as shown in Figure 4.4 which is a face-spanning subgraph of $S^{**}_{2.5}$ since it contains vertices on the boundary of every face in $S^{**}_{2.5}$. In [11], the lower bound on the number of vertices in a face-spanning subgraph is determined to be $\frac{\#f-2}{d_{out}-2}r$ Since `2.5D non-redundant peak strategy` starts on a vertex, it takes $T = \frac{\#f-2}{d_{out}-2} - 1$ time steps to complete.*

`2.5D non-redundant peak strategy` on regular triangular planar graphs

We explore the number of time steps required for execution of our strategy on regular planar graphs. A regular triangulated planar graph is a graph such that every vertex has the same out-degree, only contains triangular faces, and is planar. This small set of graphs are coincidentally the Schlegel diagrams of regular triangular polyhedrons, where a Schlegel diagram is the projection of a polytope from $\mathbb{R}^d$ to $\mathbb{R}^{d-1}$ through a point beyond one of its faces as shown in Figure 4.9. There are only three regular triangular polyhedrons: The tetrahedron, octohedron, and icosahedron.

We consider `2.5D non-redundant peak strategy` execution on any regular triangular planar graph and given them additional information reguarding the out-degree of vertices in $S^{**}_{2.5}$. We are now able to define $\mathcal{U}(v)$ as the set of undetected neighboring vertices of v. Since the agents know the out-degree of every vertex in $S^{**}_{2.5}$, they are able to deduce $|\mathcal{U}(v)|$. We ammend the exploration phase of `2.5D non-redundant peak strategy` in Algorithm 4.

Here, we improve the exploration method by prioritizing relabeling of vertices that have more undetected neighbors. With this ammendment, we achieve an interesting result in execution of `2.5D non-redundant peak strategy` on regular planar triangular graphs.

Proposition 4.3.5 *Execution of `2.5D non-redundant peak strategy` completes in a defined number of time steps with the ammendment given by Algorithm 4 on Schlegel diagrams of regular triangular polyhedrons. Execution on tetrahedon takes 1 time step. Execution on octohedrons take 2 time steps with 1 relay. Execution on icosahedrons take 5 time steps.*

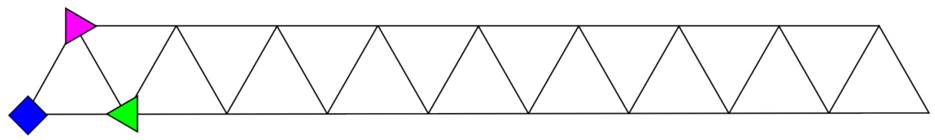
$S^{**}_{2.5G}$ after any execution of `2.5D non-redundant peak strategy` on Schlegel

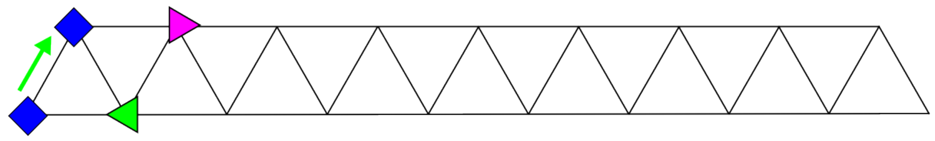
Algorithm 4 Exploration ammendment

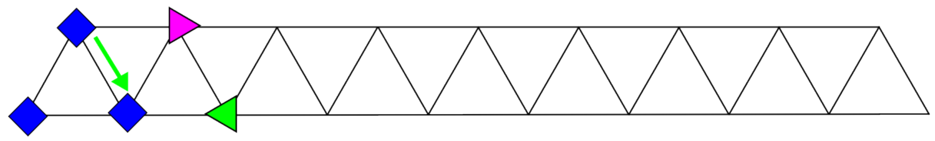
```
Explore
  if an unoccupied vertex $v \in \mathcal{D}$ of color $G_c$ exists:
    if $a$ has the lowest $ID$ that neighbors $v$
      $a$ moves to $v$
      $a$ broadcasts and updates tree
      $a$ sets *explored* to True
  else if $a$ neighbors $v \in \mathcal{D}$ of color *1* or *2* and $|\mathcal{U}(v)| = \max_{w \in \mathcal{D}} |\mathcal{U}(w)|$
    if no agents with lower $ID$ have moved this time step
      $a$ moves to $v$
      $a$ broadcasts and updates tree
      $a$ sets *explored* True
  if a child agent, $a_c$, of $a$ moved:
    $a$ moves to last occupied vertex of $a_c$
```

diagrams of regular triangular polyhedrons always contain the same pattern as shown in Figure 4.10. The execution of `2.5D non-redundant peak strategy` on all of the regular triangular polyhedrons is shown in Figure 4.11.

Chapter 4, in part, has been submitted for publication of the material as it may appear in the Dynamic Systems and Control Conference, 2016, Cortés, Jorge. The thesis author was the primary investigator and author of this paper.


(a) An initial $S^{**}_{2.5}$ (worst case) after coloring


(b) Second step of $S^{**}_{2.5}$ (worst case) after coloring


(c) Third step $S^{**}_{2.5}$ (worst case) after coloring

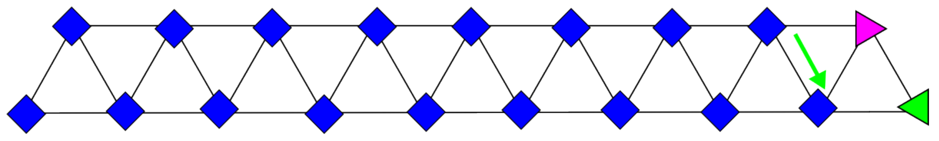
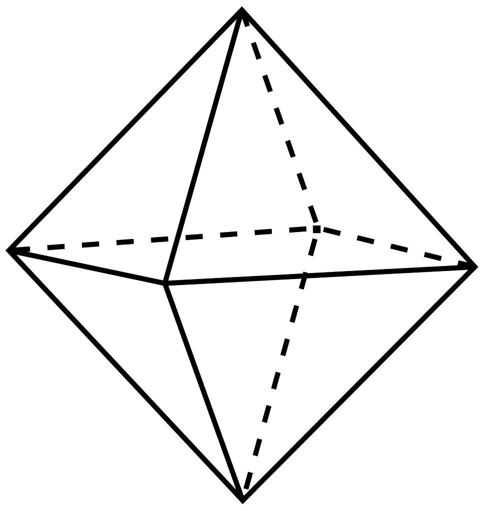

(d) Last step of $S^{**}_{2.5}$ (worst case) after coloring

Figure 4.8: A $S^{**}_{2.5}$ with the potential of taking $|V^{**}|-3$ time steps to complete.


(a) An Octohedron

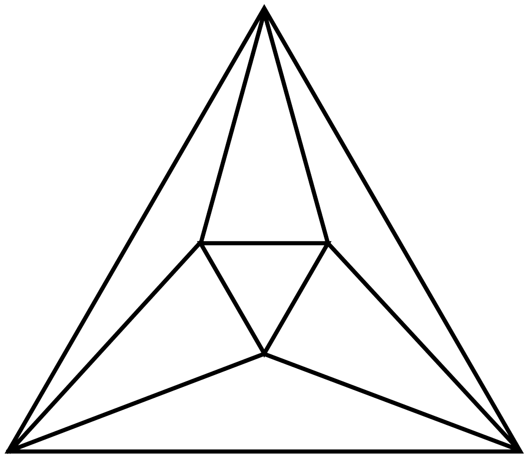

(b) Schlegel diagram of octohedron

Figure 4.9: The transformation of a regular polyhedron (octohedron) to a graph using Schlegel diagrams.

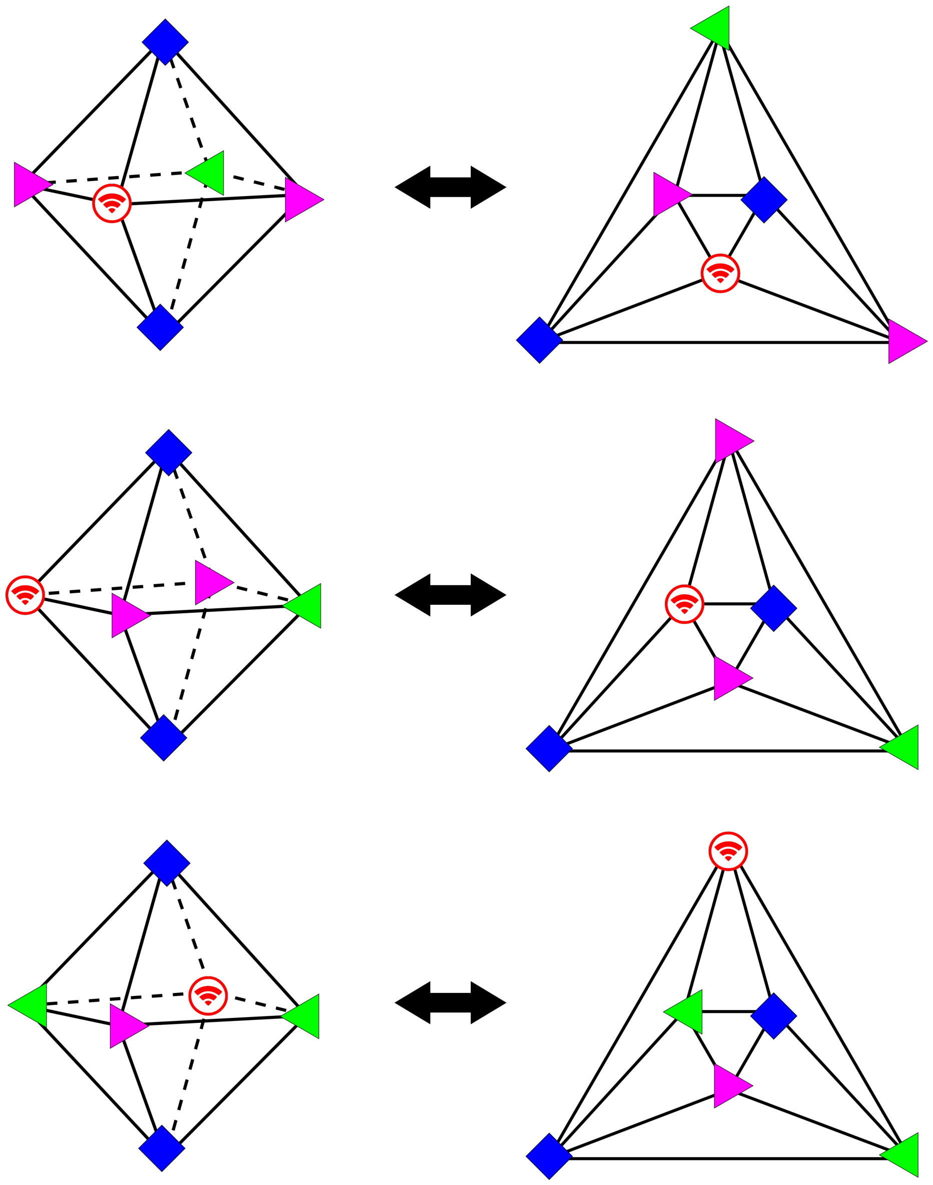


Figure 4.10: A couple varied executions of `2.5D non-redundant peak strategy` on an octohedron graph. All final configurations contain 2 agents, 1 relay, and requires 2 time steps to complete.

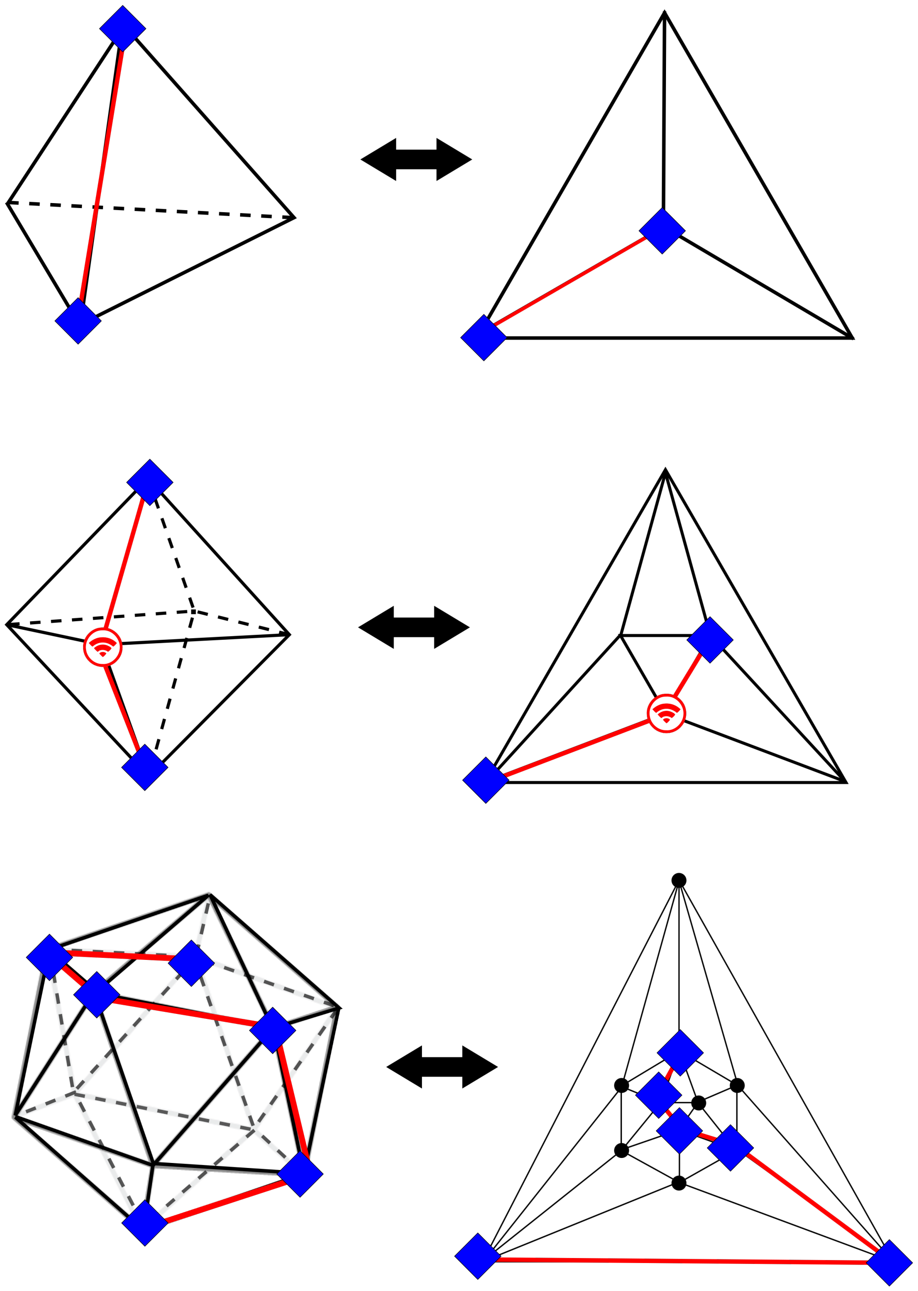


Figure 4.11: A couple varied executions of `2.5D non-redundant peak strategy` on a regular planar paths.

Chapter 5

Lab work

As part of my work towards my degree, I work in the Multi-agent Robotics (MURO) lab directed by Professor Jorge Cortés and Professor Sonia Martínez. The MURO lab focuses on implementation of distributed algorithms for robots, and research in human-swarm interaction. My duties in the lab include mentoring of undergraduate students, development of Robotic Operating System (ROS) programs, survey and implementation of algorithms to aid localization and control of multi-agent systems, and android development for human-robot interaction. The MURO lab uses both mobile ground robots(Figure) and quadcopters that interact cooperatively

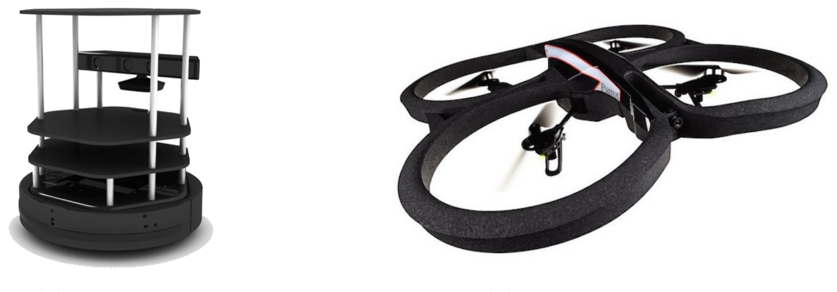

(a) Turtlebot (b) AR parrot drone 2

Figure 5.1: The mobile ground robot (Turtlebot) and aerial robot (AR parrot drone 2) models that we use in MURO lab are shown.

5.1 Robotic operating system

Robotic Operating System is a set of software libraries that aid robotic applications. The set of libraries affiliated with ROS extend from localization, control, and machine vision techniques. ROS derived its popularity from the wireless communication protocols made simple using a subscribe and publish model as shown in Figure 5.2. This communication protocol allows for implementation of practical robots that match many assumptions made in distributed robotic algorithms. For example, we are able to emulate situations where two agents are only able to communicate if visible to each other by enabling or disabling communication through ROS.

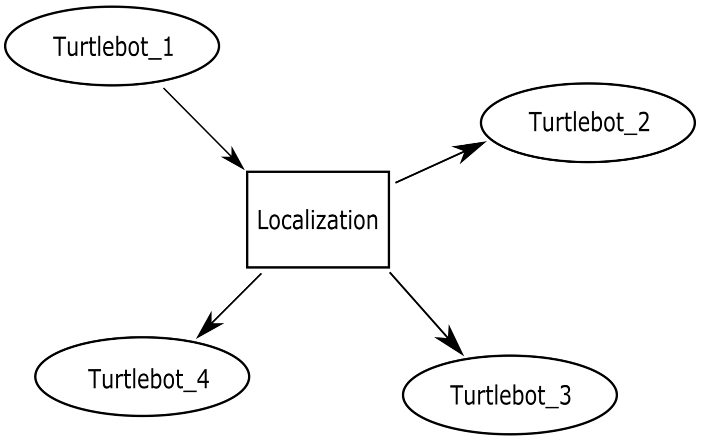


Figure 5.2: The ROS subscriber publisher model simplified. Here there are 4 physical machines communicating with each other, Turtlebot_1,Turtlebot_2,Turtlebot_3, and Turtlebot_4. ROS uses a subscriber publisher model where an agent publishes a message onto a topic. Subscribers poll to receive messages to the topics that they are subscribed to. Here Turtlebot_1 publishes to a topic called "Localization" which contains information regarding the location of Turtlebot_1. Turtlebot_1,Turtlebot_2, and Turtlebot_3 subscribed to "Localization" and receive the message that Turtlebot_1 published.

5.2 Lab implementation and Kalman filtering

While a part of the MURO lab, I have been a part of implementing and demonstrating algorithms. Such algorithms include extended Kalman filters(EKF) and control algorithms for turtlebots and AR parrot drones, cyclic pursuit, several voronoi deployments, and formation control. Figure 5.3 shows the ROS network used in the MURO lab for combining the EKF, a distributed algorithm, and controls.

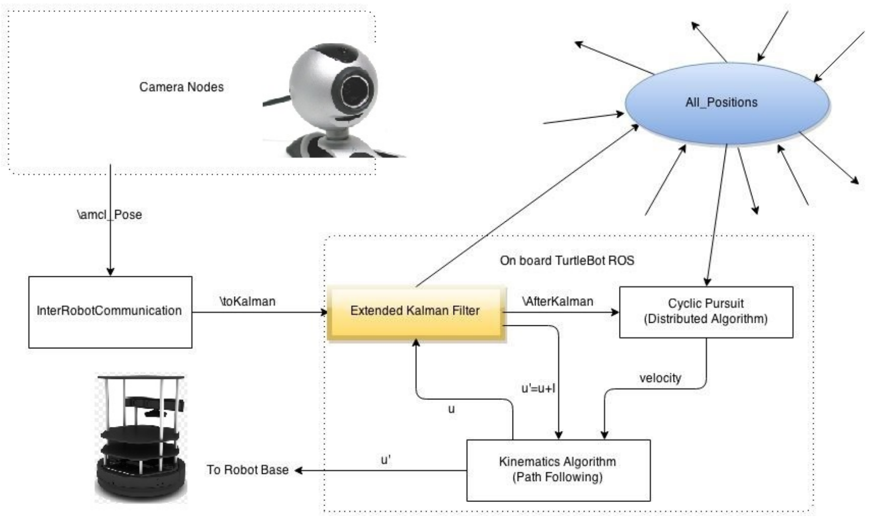


Figure 5.3: Here the ROS network in MURO lab is shown.

In order for our robots to function, we needed reliable localization. Previously, locations were published when a downward facing camera detected a turtlebot. This was a problem because the downward facing camera would often not detect the turtlebot due to lighting conditions. A fundamental solution is to implement an extended Kalman filter, which allows for more accurate state estimations and consistently published messages. We measure the location of a turtlebot, y_k and assume the measurement noise w_k is Gaussian, with a variance that we experimentally determine. For the extended Kalman filter created for turtlebots, we assume unicycle dynamics:

$$x_{1,k+1} = x_{1,k}\ u_{1,k}cos(\theta_k)$$

$$x_{2,k+1} = x_{2,k}\ u_{1,k}sin(\theta_k)$$

$$\theta_{k+1} = \theta_k + u_{2,k}$$

$$\text{Generalized as } x_{k+1} = F_k x_k + G_k u_k + w_k$$

The Kalman filter fuses the state measurement, y_k, and state prediction, $\hat{x}_{k|k-1}$, to get best state estimation, $\hat{x}_{k|k}$, as long as the spectral densities of disturbance in our state prediction and measurement error, Q and R respectively, are known. The Kalman filtering process is as follows.

0. TurtleBots Move:

$x_{k+1} = F_k x_k + G_k u_k + w_k$

$y_k = H_k x_k + J_k u_k + r_k$

1. Predict the next State, Output and Covariance:

$\hat{x}_{k+1|k} = F_k \hat{x}_{k|k} + G_k u_k$

$y_k = H_k x_k + J_k u_k$

$P_{k+1|k} = FkP_{k|k}\ F -_k^T + Q_k$

2. Incorporate new measurements (Measurement Update):

$L_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}$

$\hat{x}_{k|k} = \hat{x}_{k|k-1} + L_k(y_k - \hat{y}_{k|k-1}\ )$

$P_{k|k} = (I - L_k H_k) P_{k|k-1}$

Where H is our measurement model, F is our state model, P is the covariance matrix (a measurement of state estimate accuracy), and L is our the Kalman gain.

When assumed that no measurements are skipped, we can combine elements of the Kalman gain and covariance in one step called the Riccati difference equation (RDE).

$$P_{k+1|k} = F_k P_{k|k-1} F_k^T - F_k P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1} H_k P_{k|k-1} F_k^T + Q_k$$

$$L_k = -P_k H_k^T R_k^{-1}$$

In a time-invariant model, a solution to P exists and can be determined by the discrete Algebraic Riccati Equation (dARE).

$$L_\infty = P_\infty^- H^T (H P_\infty^- H^T + R)^{-1}$$

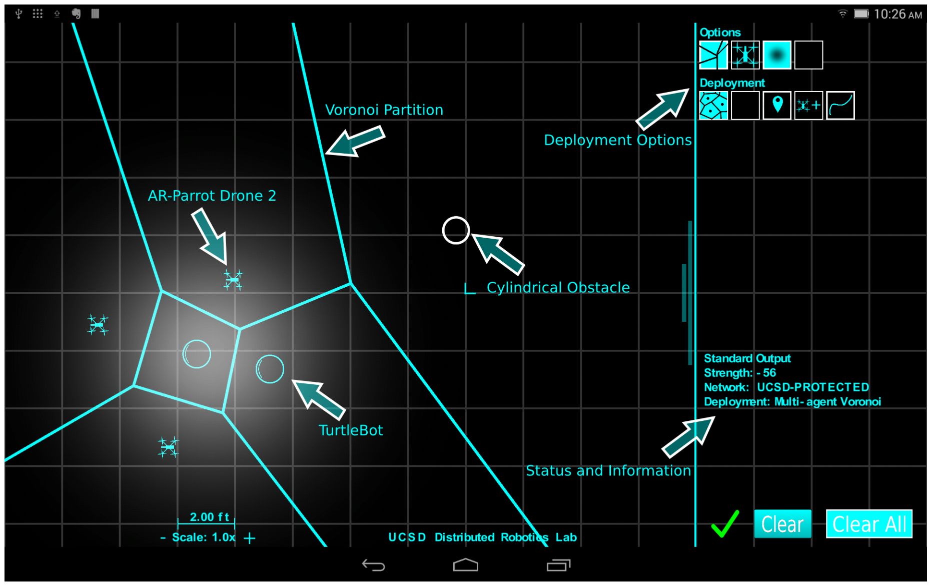
There are several state estimation methods that can be used for non-linear dynamics including particle, scented Kalman, ensemble Kalman, linearized Kalman, and the extended Kalman filter. Some of these methods, such as the particle filter are more practical when the state is near a repulsive equilibrium, and the probability density function diverges as time increases. The most commonly used non-linear filter is the extended Kalman filter, which runs under the assumptions that the state evolution is non-linear and the measurement is linear(F is non-linear, H is linear). In order to run the extended Kalman filter, F needs to be linearized when solving the Riccati difference equation.

$$F = \frac{\partial f}{\partial x}(\hat{x}_{k-1|k-1}, \hat{u}_{k-1|k-1}, 0) = \begin{bmatrix} 1 & 0 & -tv_{k-1} sin(\theta_{k-1}) \\ 0 & 1 & tv_{k-1} cos(\theta_{k-1}) \\ 0 & 0 & 1 \end{bmatrix}$$

5.3 Android

Another goal of the MURO lab is to enable human-robot interaction. With our test-bed we control small fleets of robots with high level human motivations.

We developed an application for the MURO lab on android that allows for wireless communication to our robots as shown in Figure 5.4. The application is useful for demonstrating distributed algorithms by allowing the user to switch between types of deployment, such as Lloyd's algorithm, formation control, and obstacle avoidance, and provides parameters for influencing these algorithms.



Standard Output
Strength: - 56
Network: UCSD-PROTECTED
Deployment: Multi-agent Voronoi

Figure 5.4: The android application that allows us to communicate with our fleet of robots.

Chapter 6

Final notes

6.1 Conclusion

We have explored algorithms on distributed exploration and guarding deployment for coordinated agents in 1.5D and 2.5D environments.

In the 1.5D setting, we have determined that the minimum sufficient and some times necessary number of agents required to guard the terrain is floor$(|\mathcal{P}|/2)+1$. We have developed two methods for exploration and guarding of 1.5D terrains. In both strategies, we assume that the agents are unaware of their initial location and the structure of the 1.5D terrain. The agents begin by splitting into two groups, A_+ and A_-, determined by which group of agents reach the boundary of $S_{1.5}$ first. We first introduce the more simple algorithm, `1.5D alternate peak strategy with wait`, such that agents in A_- wait to receive the "goal" flag from A_+ before continuing to deploy on $S_{1.5}$. We then introduce the second algorithm, `1.5D alternate peak strategy w/o wait`, such that agents in A_- rationalize that they belong in A_- and then continue to deploy on $S_{1.5}$. The time of execution was determined for both strategies after they are introduced.

In the 2.5D polyhedral terrain setting we begin by transforming $S_{2.5}$ to $S^*_{2.5}$, which is the result of projected $S_{2.5}$ to a plane. We incorporated visibility of $S_{2.5}$ in our treatment of $S_{2.5}$ deployment by transforming $S^*_{2.5}$ to $S^{**}_{2.5}$, which is the result of contracting visible convex hulls (reducible vertices) in $S^*_{2.5}$. We introduced the concept of guarding sets with non-redundant vertices. Combining this concept with coloring strategies for planar graphs, we have devised a distributed algorithm for simultaneous exploration and guarding. The resulting algorithm constructs a tree for communication and removes redundant vertices so that $|A| = 2|V^{**}|/3$ is sufficient. The lower and upper bounds of time of execution was determined for `2.5D non-redundant peak strategy`. The lower bound was determined by recognizing that `2.5D non-redundant peak strategy` deploys agents in a tree on $S^{**}_{2.5}$ that spans all faces, and the lower bound of vertices in a face-spanning subgraph is known. The upper bound was determined by generation of a worst case scenario as shown in Figure 4.8. Finally, the time of execution is determined for regular planar triangulated graphs, which requires that the agents know that they are deploying on a regular graph, and the out-degree of vertices is known.

6.2 Future work

In the 1.5D terrain environment, future work includes constraining agent capabilities. For example, we will explore the problem when the agents have a limited range of visibility. This is a challenging problem in which may result in a deployment strategy that utilizes peaks less than we have presented in this thesis. We would also like to consider the scenario where agents are not able to travel peak to peak at every time step, but a distance along edges in $S_{1.5}$. By removing these assumptions, we would more accurately depict a deployment in $S_{1.5}$. This would also make determination of sufficient and sometimes necessary number of agents and time of execution more complex and perhaps impossible without knowing more information about $S_{1.5}$ beforehand.

In the 2.5D terrain environment, future work also includes constraining agent capabilities. By constraining the agents to a limited ranged of visibility, Lemma 4.2 no longer holds. There are a couple possible solutions to this problem. One solution would be to transform $S_{2.5}$ into a new type of planar graph with the visibility range of agents in mind. Another solution would be to use the visibility graph of $S_{2.5}$, which is a graph where vertices are connected if they are visible to each other. However, determining an optimal guarding through the use of a visibility graph is done through centralized methods and is considered NP-Complete. Further research in our algorithm could be the determination of an algorithm that creates a face-spanning tree with no 3-cycles. If successful, such an algorithm would be a 5-coloring deployment solution for planar graphs.

In the MURO lab, future work includes the implementation of our $S_{2.5}$ deployment strategy with our test-bed. In order to do this, we plan on simulating a $S_{2.5}^{**}$ environment by projecting the graph onto the floor where the turtlebots are deployed. This requires mounting of several projectors on the ceiling of our lab. A method for distributing work among the projectors is needed as well since the height of the turtlebots may occlude projection to the floor. Once we can project $S_{2.5}^{**}$ onto the floor, we will then implement `2.5D non-redundant peak strategy` on each of the turtlebots, while supervising communication ability with the overhead camera.

Bibliography

[1] J. O'Rourke, *Art Gallery Theorems and Algorithms.* Oxford University Press, 1987.

[2] V. Chvátal, "A combinatorial theorem in plane geometry," *Journal of Combinatorial Theory. Series B*, vol. 18, pp. 39–41, 1975.

[3] F. Hurtado, M. Loffler, I. Matos, V. Sacristan, M. Saumell, R. I. Silveria, and F. Staals, "Terrain visibility with multiple viewpoints," in *International Symposium on Algorithms and Computation*, (Jeonju, Korea), pp. 317–327, 2014.

[4] S. Friedrichs, M. Hemmer, and C. Schmidt, "A PTAS for the continuous 1.5d terrain guarding problem," in *Canadian Conference on Computational Geometry*, (Halifax, Nova Scotia, Canada), Aug. 2014. Electronic Proceedings.

[5] M. Gibson, G. Kanade, E. Krohn, and K. Varadarajan, "Guarding terrains via local search," *Journal of computational geometry*, vol. 5, no. 1, pp. 168–178, 2014.

[6] K. Appel and W. Haken, "Every planar map is four colorable. Part i: Discharging," *Illinois J. Math*, vol. 21, pp. 429–490, 1977.

[7] P. Bose, T. Shermer, G. Toussaint, and B. Zhu, "Guarding polyhedral terrains," *Computational Geometry*, vol. 7, pp. 173–185, 1997.

[8] N. Chiba, T. Nishizeki, and N. Saito, "A linear 5-coloring algorithm of planar graphs," *Journal of Algorithms*, vol. 2, pp. 317–327, 1981.

[9] M. H. Williams, "A linear algorithm for colouring planar graphs with five colours," *The Computer Journal*, vol. 28, pp. 78–81, 1985.

[10] N. Robertson, D. P. Sanders, P. Seymour, and R. Thomas, "Efficiently four-coloring planar graphs," in *ACM Symposium on Theory of Computing*, (Philadelphia, PA), pp. 571–575, 1996.

[11] M. Patwary and S. Rahman, "Minimum face-spanning subgraphs of plane graphs," *AKCE International Journal of Graphs and Combinatorics*, vol. 7, no. 2, pp. 133–150, 2010.

[12] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks.* Applied Mathematics Series, Princeton University Press, 2009. Electronically available at `http://coordinationbook.info`.

[13] A. Ganguli, J. Cortés, and F. Bullo, "Distributed deployment of asynchronous guards in art galleries," in *American Control Conference*, (Minneapolis, MN), pp. 1416–1421, June 2006.

[14] H. Grötzsch, "Ein dreifarbensatz für dreikreisfreie netze auf der kugel," *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihey*, vol. 8, pp. 109–120, 1959.