Proceedings of ASME 2016 Dynamic Systems and Control Conference
DSCC 2016
October 12-14, 2016, Minneapolis, Minnesota, USA

DSCC2016-9820

VISIBILITY-BASED DISTRIBUTED DEPLOYMENT OF ROBOTIC TEAMS IN POLYHEDRAL TERRAINS

Aaron Ma **Jorge Cortés** *

ABSTRACT

This paper presents deployment strategies for a team of multiple mobile robots with line-of-sight visibility in 1.5D and 2.5D terrain environments. Our objective is to distributively achieve full visibility of a polyhedral environment. In the 1.5D polyhedral terrain, we achieve this by determining a set of locations that the robots can distributively occupy. In the 2.5D polyhedral terrain, we achieve full visibility by simultaneously exploring, coloring, and guarding the environment.

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.

Literature review

This paper 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). Our paper also builds on notions from distributed robotic networks [11] and distributed deployment of mobile robots to guard art galleries based on visibility and line of sight [12].

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 authors are with the Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA 92093, USA, {aam021,cortes}@ucsd.edu

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 paper illustrate our results. Some proofs are omitted for reasons of space and will appear elsewhere.

2 Preliminaries

This section introduces basic notation, planar graphs and coloring, and polyhedral terrains.

2.1 Notation

Given a set S, we let $|S|$ denote its cardinality. We let 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 $y^p = 0$ 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].

2.3 Polyhedral terrains in 1.5D and 2.5D

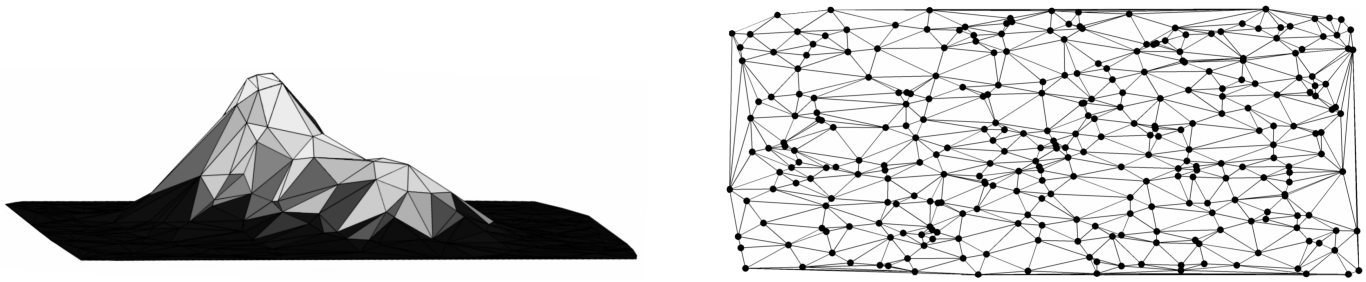
1.5D and 2.5D polyhedral terrains correspond to the graphs of continuous piecewise affine functions on $\mathbb{R}$ and $\mathbb{R}^2$, respectively. Formally, given a continuous piecewise affine function $f : I \subset \mathbb{R} \to \mathbb{R}$, with I an interval, its associated 1.5D terrain is $S_{1.5}(f) = \{(x, f(x)) : x \in I\} \subset \mathbb{R}^2$. 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 $x^v_{i-1} < x^v_i < x^v_{i+1}$ 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 $\min(x^v_1, x^v_2) < x^v < \max(x^v_1, x^v_2)$ (resp. $\min(x^v_1, x^v_2) \le x^v \le \max(x^v_1, x^v_2)$). 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 1 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

$$s_{\overline{v_1 v_2}} > s_{\overline{v_1 w}}, \quad \forall w \in K_{v_1 v_2}. \tag{1}$$



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

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

Figure 1. The 2.5D terrain converted into a planar graph.

If the test is passed, then the vertices are visible to each other. In $S_{1.5}$, $K_{v_1v_2} = \mathcal{J}_{(v_1,v_2)}$ is the set of vertices between v_1 and v_2. In $S_{2.5}$, $K_{v_1v_2}$ is the set of points on $S_{2.5}$ that share x and y-coordinates with $\overline{v_1v_2}$. 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 $Q(V_w) = V$.

3 Problem statement

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 section we describe in detail the model for the robotic network and its capabilities. Given a set of agents, A, Each individual 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 omnidirectional 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.

4 Distributed deployment over 1.5D terrains

This section 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.

4.1 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 4.1. *(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. 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 $s_{v_1v} > s_{v_1k}$ for all $v \in \mathcal{J}_{(v_1,v_2)}$ and $k \in K_{v_1v} = \mathcal{J}_{(v_1,v)}$. Hence, all vertices between v_1 and v_2 are visible from either v_1 or v_2. □

As a consequence of Lemma 4.1, 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 4.2. *(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. From Lemma 4.1, 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. □

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 4.3. *(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. The fact that $Q(G) = V$ readily follows from Proposition 4.2. 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. □

4.2 `1.5D alternate peak strategy`

Given our analysis in Section 4.1, 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). Depending on the location of v_0 within the environment, one of these two sets contains 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 4.4. *(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. •

Remark 4.5. *(Ordering of agents):* Agents occupy a peak only if no other agent with lower ID occupies the same peak. As the algorithm executes, the agents naturally order themselves within their respective groups of A_- and A_+ in decreasing order of ID from v_0 in the direction they are initialized. This enables the agents to rationalize 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 $2A_- + \text{ID} - 2$ time steps. •

Figure 2 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

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

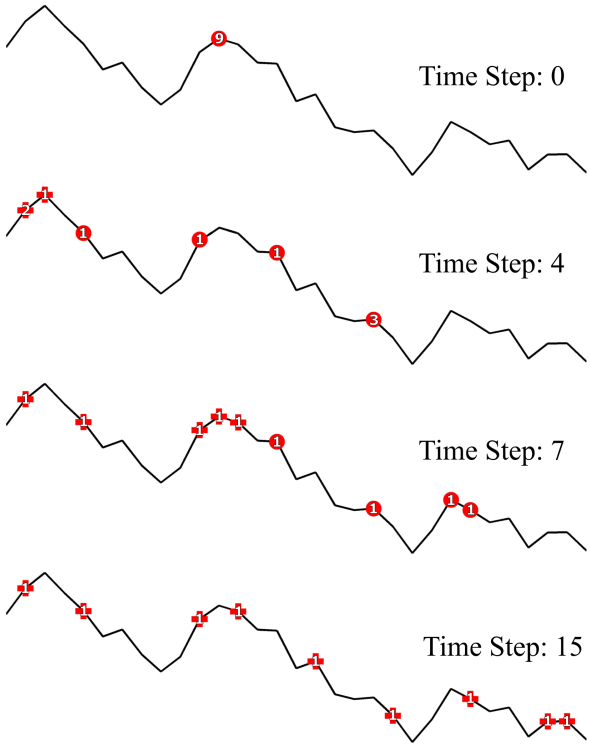


Figure 2. Execution of `1.5D alternate peak strategy with wait` on a 1.5D environment with 16 peaks. From Theorem 4.3, $|A| = 9$ agents are sufficient to achieve full visibility. All agents begin at the same initial location, v_0, of index 6 with respect to $\mathcal{P}$, and split into two groups. The location of agents with the "goal" flag not raised are shown by a red dot ❸ where the number states the number of agents on that vertex. Agents with a raised "goal" flag are denoted with a cross ✚.

agents and begins to wait for the "goal" flag. By time step: 15, the network has completely deployed achieving full visibility of the environment.

4.3 Time steps for algorithm completion

In this section 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

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

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

$$\begin{aligned} \mathcal{A} = \{v_0 \,|\, &\text{if } |A| \text{ even, } i \leq |\mathcal{P}|/6+1 \text{ or } i \geq 5|\mathcal{P}|/6-1 \\ &\text{and if } |A| \text{ odd, } i \leq |\mathcal{P}|/6 \text{ or } i \geq 5|\mathcal{P}|/6\}, \\ \mathcal{B} = \{v_0 \,|\, &\text{if } |A| \text{ even, } |\mathcal{P}|/6+1 < i < 5|\mathcal{P}|/6-1 \\ &\text{and if } |A| \text{ odd, } |\mathcal{P}|/6 < i < 5|\mathcal{P}|/6\}, \end{aligned}$$

and region $\mathcal{C}$, which only exists if $|\mathcal{P}|$ is odd and corresponds to $i = \frac{|\mathcal{P}|+1}{2}$. Figure 3 illustrates these three cases. We are ready to characterize the time complexity of the strategy with wait.

Theorem 4.6. *(`1.5D alternate peak strategy with wait` completion time): The number of time steps required by the `1.5D alternate peak strategy with`*

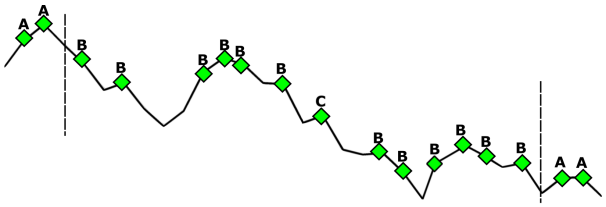


Figure 3. Illustration of cases $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ for the locations of the common initial condition of the agents. The 1.5D environment $S_{1.5}$ has $|\mathcal{P}| = 17$.

`wait` *to complete is*

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

Proof. 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, $i^* = i$, $A_{\text{lft}} = A_+$, and $A_{\text{rght}} = A_-$. 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.,

$$i^* \leq \frac{2}{3}|A_-| + 1. \tag{4}$$

Case $\mathcal{A}$: One can see that equation (4) 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 (4) 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 4.6, 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 4.7. *(`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*

$$T = \begin{cases} |\mathcal{P}| - i^* & v_0 \in \mathcal{A}, \\ \frac{7|\mathcal{P}|}{8} - \frac{i^*}{4} & v_0 \in \mathcal{B}, \\ \frac{3(|\mathcal{P}|-1)}{4} & v_0 \in \mathcal{C}. \end{cases} \tag{5}$$

5 Distributed deployment over 2.5D terrains

This section studies the distributed deployment problem over 2.5D polyhedral terrains. We introduce the concept of a 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.

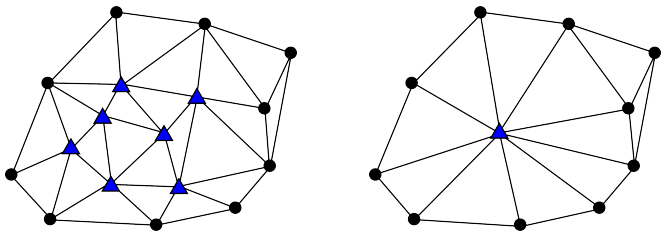
5.1 Guarding set via non-redundant vertices

$S_{2.5}$ 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 section 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. The vertices that remain in $S^{**}_{2.5}$ are then V^{**}.

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

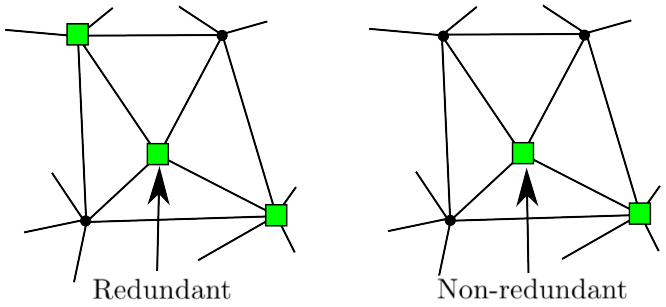
Definition 5.1. *(Redundant vertex): Consider a vertex, v, that is occupied in guarding set G. Define Δ_v as the set of triangles, t, in contact with v as determined by vertices and edges in $S^{**}_{2.5}$. v is a redundant vertex if there is another occupied vertex that is able to see* t *for all* $\mathrm{t} \in \Delta_v$*. 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 $Q(G) = Q(G\backslash v)$.*



(a) $S^{*}_{2.5}$ (b) $S^{**}_{2.5}$

Figure 4. Example of a reducible set in $S^{*}_{2.5}$ being contracted. The symbol ▲ represents vertices in $\mathcal{R}$. $|\mathcal{R}| = 7$ is reduced to $|\mathcal{R}| = 1$ after contraction.

Figure 5 provides an illustration of the concept of redundant vertex.



(a) G with a redundant vertex (b) G with a non-redundant vertex

Figure 5. Illustration of the concept of redundant vertex. Green squares correspond to vertices in the guarding set G. In (a), a vertex that does not uniquely guard neighboring triangles is redundant. In (b), the same vertex uniquely guards a neighboring triangle, and is therefore non-redundant.

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

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 5.2.

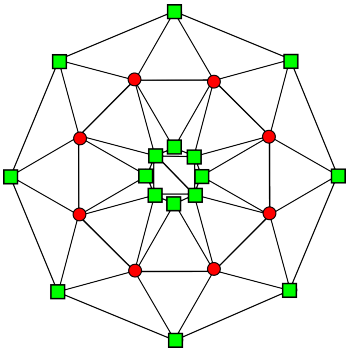


Figure 6. The configuration that contains the maximum number of non-redundant vertices in $S^{**}_{2.5}$. Here, the green squares are in G and are non-redundant. The red circles represent vertices that cannot be added to G without creating redundant vertices.

5.2 $S_{2.5}$ exploration and guarding algorithm

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 $\{1,2,G_c\}$, where G_c is a label for a color in $\{3,4,5,6\}$ 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 5.3. *(Three-coloring of a triangle): Every triangular face in $S^{**}_{2.5}$ contains a vertex labeled a color in G_c after coloring.*

Occupying all vertices that are not $\{1,2\}$ 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 5.4. *(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 $S^{**}_{2.5}$ andD. •

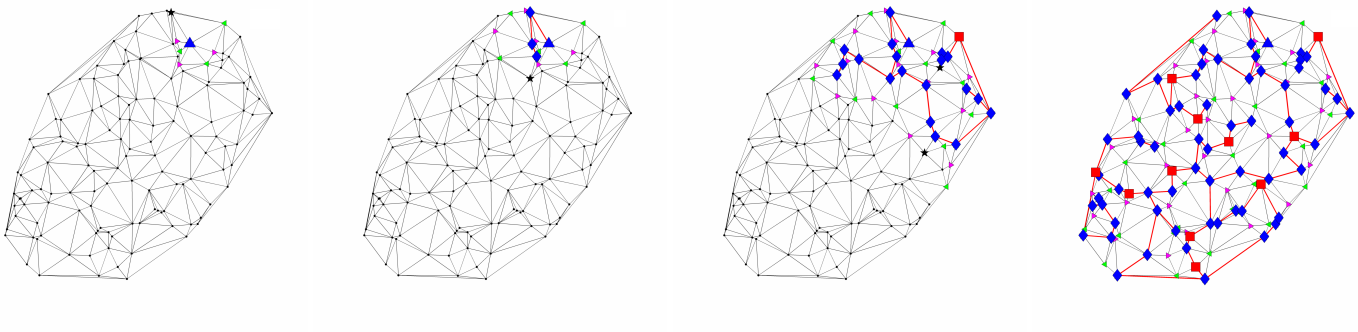
Lemma 5.5. *(Completion of the `2.5D non-redundant peak strategy`): The exploration process results in complete exploration of $S_{2.5}$.*

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.

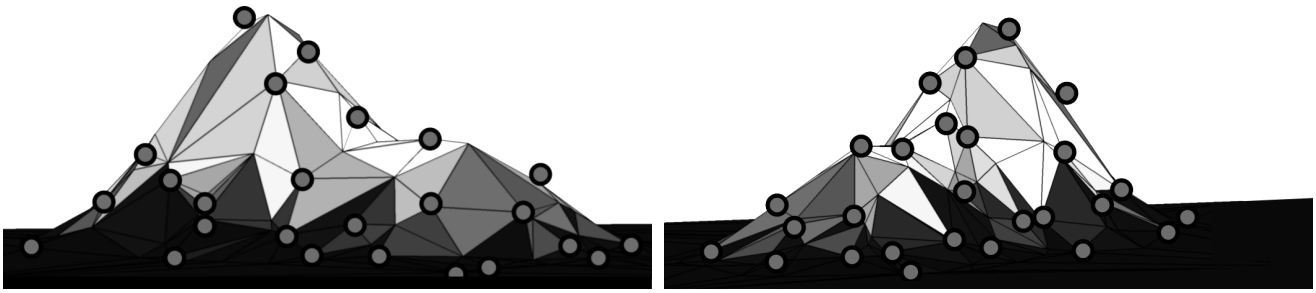
Remark 5.6. *(`Redundant agent removal`):* By construction, this process guarantees $|Q|$ never decreases. It is necessary that Algorithm 3 be recursive because the set of redundant agents changes every time a redundant agent is removed. •

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 7. 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 ▲. The vertices that are visible to the agents are denoted with ●. The agents label the newly detected vertices surrounding v_0 with colors priority: $1 =$ ◀, $2 =$ ▶, and $G_c =$ ★. 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 1 and the shift of the robots is represented by a black arrow. Notice that in $\mathcal{D}$, there are no vertices with color G_c. The agents must change the color of a vertex in $\mathcal{D}$ with color 1 or 2 to G_c. 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 1, 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.



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

Figure 8. Results of `2.5D non-redundant peak strategy` in $S_{2.5}$ are shown.

Theorem 5.7. *(The `2.5D non-redundant peak strategy` completion time): The $S_{2.5}$ deployment strategy takes at most $|V^{**}|-3$ time steps to complete.*

6 Conclusions

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 a method for both exploration and guarding of the 1.5D terrain. In the 2.5D setting, we have 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. Future work will explore the extension of our algorithm design to scenarios where agents have limited visibility regions, such as those determined by limited ranges or cone-like shapes, the consideration of other motion models for the agents that allow them to traverse across the planes of a 2.5D terrain instead of just the edges, and the implementation of our results on an experimental testbed with a fleet of quadrotors.

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

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 exclusive 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

Acknowledgments

This work was supported in part by Northrop Grumman through seed funding of the UCSD Contextual Robotics Institute.

REFERENCES

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

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

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

[4] Friedrichs, S., Hemmer, M., and Schmidt, C., 2014. "A PTAS for the continuous 1.5d terrain guarding problem". In Canadian Conference on Computational Geometry. Electronic Proceedings.

[5] Gibson, M., Kanade, G., Krohn, E., and Varadarajan, K., 2014. "Guarding terrains via local search". *Journal of Computational Geometry,* ***5***(1), pp. 168–178.

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

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

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

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

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

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

[12] Ganguli, A., Cortés, J., and Bullo, F., 2006. "Distributed deployment of asynchronous guards in art galleries". In American Control Conference, pp. 1416–1421.