Online Exploration of an Unknown Region of Interest with a Team of Aerial Robots

Yoonchang Sung[1], Deeksha Dixit[2] and Pratap Tokekar[2]

Abstract

In this paper, we study the problem of exploring an unknown Region Of Interest (ROI) with a team of aerial robots. The size and shape of the ROI are unknown to the robots. The objective is to find a tour for each robot such that each point in the ROI must be visible from the field-of-view of some robot along its tour. In conventional exploration using ground robots, the ROI boundary is typically also as an obstacle and robots are naturally constrained to the interior of this ROI. Instead, we study the case where aerial robots are not restricted to flying inside the ROI (and can fly over the boundary of the ROI).

We propose a recursive depth-first search-based algorithm that yields a constant competitive ratio for the exploration problem. Our analysis also extends to the case where the ROI is translating, *e.g.*, in the case of marine plumes. In the simpler version of the problem where the ROI is modeled as a 2D grid, the competitive ratio is $\frac{2(S_r+S_p)(R+\lfloor\log R\rfloor)}{(S_r-S_p)(1+\lfloor\log R\rfloor)}$ where R is the number of robots, and S_r and S_p are the robot speed and the ROI speed, respectively. We also consider a more realistic scenario where the ROI shape is not restricted to grid cells but an arbitrary shape. We show our algorithm has $\frac{2(S_r+S_p)(18R+\lfloor\log R\rfloor)}{(S_r-S_p)(1+\lfloor\log R\rfloor)}$ competitive ratio under some conditions. We empirically verify our algorithm using simulations as well as a proof-of-concept experiment mapping a 2D ROI using an aerial robot with a downwards-facing camera.

1 Introduction

We investigate the problem of exploring and mapping an unknown 2D Region Of Interest (ROI) using a team of autonomous Unmanned Aerial Vehicles (UAVs). Our overall vision is to develop aerial coverage algorithms for enabling a team of UAVs to assist emergency responders in disaster scenarios or environmental scientists in data collection. Figure 1 shows one motivating scenario where a UAV can be used to map the region in a lake that is contaminated by a leaked pollutant (*e.g.*, chemical spill in a lake). The size and shape of the contaminated region is usually not known until observed by the UAVs. If the UAVs are flying at lower altitudes or if the contaminated region is large, the UAV will need to plan its motion to explore and map out the unknown ROI using its onboard sensors (*e.g.*, downwards-facing camera in the case of visible contaminants such as chemical spills).

Another example is assessing the damage to structures after a natural disaster (Fernandez Galarreta et al. (2015)) using downwards-facing cameras mounted on UAVs. Such assessment is required to estimate the cost of recovery and reconstruction. As in the previous example, the exact shape of the region that has undergone damage may not be known a priori and must be mapped using the UAVs. There are many such examples of aerial mapping of ROIs of unknown shape and sizes using UAVs equipped with appropriate sensors.

In such scenarios, the larger environment which contains the ROI is known but the exact shape of the ROI is unknown. We assume that the UAV has a sensing pipeline that is capable of taking in the downwards-facing image and distinguishing whether the UAV is above an ROI or

Figure 1. A UAV conducting plume exploration in an abandoned quarry near Blacksburg, Virginia.

not. The goal is to plan a trajectory for a team of UAVs to collectively map the ROI in the least amount of time. The problem of exploring an unknown 2D environment is a well-studied one in the robotics (Brass et al. (2011); Juliá

[1]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA
[2]Department of Computer Science, University of Maryland, College Park, MD, USA
Most of the work was done when the authors were affiliated with the Department of Electrical and Computer Engineering at Virginia Polytechnic Institute and State University, Blacksburg, VA, USA.

Corresponding author:
Yoonchang Sung, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 32 Vassar Street, Cambridge, MA 02139, USA.
Email: yooncs8@csail.mit.edu

et al. (2012); Nuske et al. (2015); Girdhar et al. (2014)) and computational geometry (Icking et al. (2005); Kolenderska et al. (2009); Fraigniaud et al. (2006); Higashikawa et al. (2014)) communities. However, the problem considered in this paper differs from these works in two critical ways.

In our setup the robots are not restricted to stay inside (or over) the ROI all the time. The robots can fly over locations that are not part of the ROI, thereby allowing them to "shortcut" from one part of the ROI to the other (Figure 2). The UAVs fly at an altitude higher than the 2D plane containing the ROI. Contrast this with conventional 2D exploration problems, where the robots are restricted to stay within the boundary of the environment. Because the robots do not know the shape of the ROI a priori, they may not be able to take a "shortcut" even if one exists. As a result, the robots may end up taking a longer path, resulting in a poorer performance. Nevertheless, we present an algorithm that is *competitive* with respect to the optimal algorithm.

While the focus is on monitoring static scenes, our work also extends to translating ROI. In that case, we show that the performance of the algorithm is, not surprisingly, a function of the relative speeds of the UAV and the ROI. Depending on the application, there may exist multiple ROIs in the environment. Furthermore, the UAV may start in a region that is not part of the ROI and may have to find the ROI in the first place. In such cases, we can use a boustrophedon search pattern to find the ROI (Tokekar et al. (2013)). In this paper, we focus only on mapping a single connected ROI. The single ROI algorithm can be extended to multiple static ROIs.

We use the notion of *competitive ratio* (Borodin and El-Yaniv (2005)) to analyze the performance of our algorithm. The competitive ratio for an online algorithm is defined as the largest (*i.e.*, worst-case input) ratio of the time taken by the online algorithm to the time taken by an optimal offline algorithm. The offline algorithm is one which knows the shape of the 2D ROI a priori. We seek algorithms that have a low (preferably, constant) competitive ratio. Our main result is a constant competitive ratio for exploring a translating ROI for a fixed number of robots. The constant depends on the relative speeds of the ROI and the robots.

We require the robots to ensure that all points of the ROI are within the sensor footprints of at least one of the aerial robots along their paths. The objective is to minimize the time required for all the robots to explore the ROI and return back to the starting position. Our algorithm builds on the one presented by Higashikawa et al. (2014) for exploring an unknown binary tree. We show how to reduce the problem of exploring the ROI to that of exploring a binary tree. We first start with the simpler scenario where the ROI is modeled as a 2D grid and then extend it to translating ROI. We further generalize it to the case where the ROI boundary is any smooth (formally defined in Section 3) 2D curve with a finite sensor footprint binary sensor. For both cases, we show that our algorithm yields a constant-competitive ratio.

We validate our algorithm through simulations that quantify the performance as a function of the size of the ROI, the number of robots, and the relative speeds of the ROI and the robots. We also conduct a proof-of-concept field experiment using a UAV with a downwards-facing camera to explore and map a stationary region of interest (runway). We

discuss how to implement the algorithm in a practical setting and discuss challenges associated with noisy measurements.

In summary, the contributions of the paper are as follows:

- We propose a new exploration problem for aerial robots where the robots are not restricted to lie within the boundary of the region of interest being mapped.
- We present a constant competitive ratio algorithm for this problem. Our analysis allows for arbitrarily shaped ROIs as well as possibly translating ones.
- In addition to theoretical results, we evaluate our algorithm through simulations and proof-of-concept demonstrations.

A preliminary version of this paper was presented in Sung and Tokekar (2019). This version improves upon Sung and Tokekar (2019) with a more expansive literature survey, a more detailed explanation on the proposed algorithm, and new simulation results and proof-of-concept experiments, including a description of how to implement the proposed algorithm using a robot with a downward-facing camera.

The rest of the paper is organized as follows. We begin by introducing the related work in Section 2. We describe the problem setup in Section 3. Our proposed algorithm for a grid-based map is presented in Section 4. We then extend this to arbitrarily shaped ROIs in Section 5. We present results from representative simulations in Section 6 and field experiments in Section 7, respectively, before concluding with a discussion of future work in Section 8.

2   Related Work

Environmental monitoring has extensively been studied in robotics due to its practical applications. Some of highlighted tasks include precision agriculture (Tokekar et al. (2016); Das et al. (2015a)), wildlife habitat monitoring (Tokekar et al. (2010, 2013); Plonski et al. (2017)) and atmospheric plume tracking (Ishida et al. (2012); Lochmatter et al. (2013); Fahad et al. (2017)). For survey results, see Dunbabin and Marques (2012). The area coverage and exploration are crucial for environmental monitoring as a given environment must be explored by robots in order to detect a target of interest. Galceran and Carreras (2013) listed coverage path planning algorithms that can be used for different sensing and motion models. In case of ROI exploration, the aim is to explore and map an ROI by robots with limited sensing capability.

The objective of online exploration (Juliá et al. (2012); Nuske et al. (2015); Girdhar et al. (2014)) is to explore and map a region without having prior knowledge on the size and shape of the region. Corah and Michael (2019) studied informative exploration in which the objective is to maximize the information gathered along the planned trajectories. They developed a near-optimal distributed algorithm for multi-robot exploration, which approximates the well-known sequential greedy assignment (Singh et al. (2009)). Plonski et al. (2017) proposed algorithms for tracking radio-tagged invasive fish using USVs and ground robots. They proved competitive ratios for navigating an environment containing an unknown obstacle and energy-efficient solar exploration. Hitz et al. (2014) focused on localizing interesting areas in an unknown environment using level set estimation to monitor

hazardous cyanobacteria blooms in lakes. The objective in these works was not to completely map an unknown environment (which is the case in this paper) but to maximize information gain, track and localize targets of interest.

Sim and Little (2009) proposed a vision-based exploration and mapping solution for a single robot. Cesare et al. (2015) developed a multi-robot exploration algorithm for heterogeneous robots with limited communication and battery-life constraints. However, these approaches do not guarantee complete coverage.

Bender et al. (2002) and Das et al. (2007) addressed the problem of dealing with unlabelled (*i.e.*, anonymous) vertices when exploring an unknown graph. The former defined a pebble that can identify a vertex and found the number of pebbles required to map an unknown environment. While the former considered the case of a single robot, the latter proposed a distributed version, allowing multiple robots to start from different vertices, and proved upper bounds on the time complexity of their algorithm. Their algorithms, however, do not yield a competitive ratio used as a performance measure in this paper.

When an ROI region (or any monitoring object) can be represented by a grid polygon, there exists literature which explores a polygonal region not only completely but also competitively with respect to the optimal trajectory. This can be categorized into *lawn mowing* and *milling* where the former allows a robot to move outside the boundary of a polygon whereas the latter does not (see Figure 2). Icking et al. (2000) proposed a strategy of generating a competitive tour for online milling which may contain holes. Icking et al. (2005) showed $\frac{4}{3}$–competitive algorithm for online milling without considering holes. The algorithms presented by Arkin et al. (2000) have $(3 + \epsilon)$–approximation for offline lawn mowing and 2.5–approximation for offline milling. Kolenderska et al. (2009) developed an online milling algorithm of a grid polygon without holes that has a competitive ratio of $\frac{5}{4}$. However, aforementioned works did not take into account a multi-agent perspective. Although Arya et al. (2001) presented an approximation algorithm for milling where multiple robots can be deployed, their algorithm solves an offline problem. In this work, we pose an online milling version for multiple robots, taking into account their limited sensor footprint.

Previous works in computational geometry assumed specific properties of the region under exploration to ease the analysis. We restrict the ROI to satisfy a specific notion of fatness (defined in the next section). van der Stappen and Overmars (1994) used the notion of k–fatness in motion planning with obstacles — the smaller the value of k, the fatter the obstacle. Efrat (2005) defined a (α, β)–covered object if each angle of a triangle fully inside the object is at least α and each edge of this triangle is at least β multiplied by the diameter of the object are satisfied. Aloupis et al. (2014) adopted the same notation of the fatness for the application of triangulating and guarding polygons. Lee et al. (2016) used a similar fatness for a triangulation of a planar region for multi-robot coverage. These approaches exploited the fatness to prove the space complexity of their algorithms. In this work, we also define the fatness for proving the competitive ratio for arbitrary ROI shape.
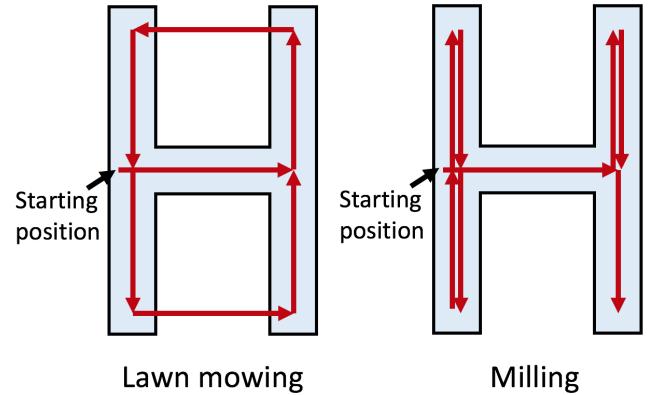

Figure 2. Example of different trajectories obtained by applying lawn mowing and milling approaches to the H-shaped ROI. Whereas lawn mowing allows the robot to move outside the ROI, milling restricts the robot to stay inside the ROI.

When multiple robots are considered, most of works (Fraigniaud et al. (2006); Brass et al. (2011); Higashikawa et al. (2014); Mahadev et al. (2017); Dynia et al. (2007); Preshant et al. (2016)) have studied a tree-based exploration by employing a recursive Depth-First Search (DFS). In these works, the environment to be explored was assumed to be a tree. Fraigniaud et al. (2006) proposed a tree exploration algorithm using R robots that is $\mathcal{O}(\frac{R}{\log R})$–competitive. In their work, each robot was allowed to observe the incident edges but not the adjacent vertices. Brass et al. (2011) used the same sensing model and improved the competitive ratio of Fraigniaud et al. (2006) to $2|E|/R + \mathcal{O}((R + r)^{R-1})$, where $|E|$ and r denote the number of edges and radius of the graph, respectively. Dynia et al. (2007) improved the lower bound proposed by Fraigniaud et al. (2006) of $2 - \frac{1}{R}$ to $\Omega(\frac{\log R}{\log \log R})$.

Megow et al. (2012) showed that the competitive ratio of a single-robot DFS is $2(2 + \epsilon)(1 + 2/\epsilon)$, where ϵ is a fixed positive parameter, when applied to general graphs. Higashikawa et al. (2014) presented a $\frac{R + \lfloor \log R \rfloor}{1 + \lfloor \log R \rfloor}$–competitive algorithm for exploring a binary tree with R robots. Das et al. (2015b) presented an algorithm for minimizing the number of robots given limited energy E for each robot.

Preshant et al. (2016) showed that the competitive ratio remains largely the same, $\frac{2(\sqrt{2}R + \log R)}{1 + \log R}$, where the environment was an orthogonal polygon* but was modeled as a tree. We build on this and generalize this to the case where the environment boundary is not necessarily orthogonal. In fact, it can be curved and may contain holes as well. Furthermore, we show how to adapt this algorithm to the case where the environment itself is translating.

To share information among multiple robots, global or local communication can be used. Das et al. (2007) and Brass et al. (2011) introduced bookkeeping devices to write local information on the vertex so that other robots can read this information when they visit the same vertex later. Lee et al. (2016) proposed distributed online exploration algorithms assuming a fully connected network. In Higashikawa et al.

*An orthogonal polygon is one in which the edges are aligned with either the X or Y axes.

(2014), robots can communicate with each other when they meet at the same vertex. We adopt the same model.

3 Problem Description

We consider the problem of mapping an ROI (Definition 1) using a team with R robots. The size and shape of the ROI are not known to the robots a priori. We use $P \in \mathbb{R}^2$ to denote the 2D ROI. Let $int(P)$ be the interior of P and ∂P be the boundary of P.

We assume that the aerial robots fly at a fixed altitude in a plane parallel to the ROI. We assume that each robot has a downwards-facing camera that yields a square footprint on the plane containing the ROI. Without loss of generality, we assume that the side length of the square sensor footprint is 1 in this work.

To avoid complications due to the trivial case of a small ROI, we assume that the ROI is at least as large as the sensor footprint of the robots. Specifically, we require the ROI to satisfy the following assumption.

Definition 1. Fat ROIs. *For any $p' \in \partial P$, let $p \in int(P)$ be a point on the normal to ∂P at p' such that p is at a distance of $\frac{\sqrt{2}}{2}$ from p'. Let $B(p)$ be an open ball of radius $\frac{\sqrt{2}}{2}$, i.e., $B(p) = \{q \mid \|p - q\|_2 < \frac{\sqrt{2}}{2}\}$ where $q \in \mathbb{R}^2$. We say that the ROI P is* fat *if $B(p)$ lies completely inside $int(P)$ for all $p' \in \partial P$.*
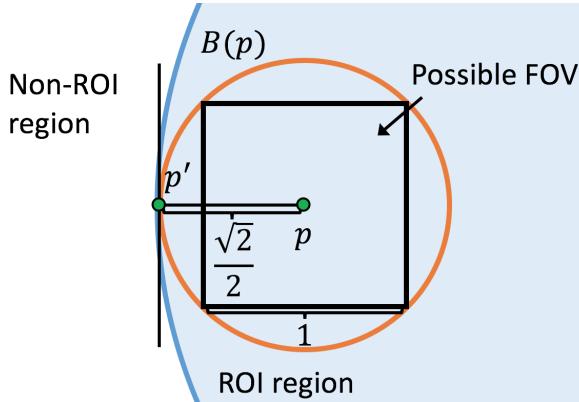

Figure 3. We restrict our attention to ROIs that are *fat* (Definition 1).

Figure 3 shows an example of an ROI that is *fat*. This definition disallows ROIs that have a *width* less than that of the sensor footprint of the robot. Note, however, we still allow the ROI to contain one or more holes.

While our focus is on mapping ROIs that are stationary, our analysis also extends to the case where the ROI is translating. For that case, we assume that the ROI translates with a fixed speed of S_p in a fixed direction, both of which are known to the robots.[†] For example, the velocity of a plume can be determined from the flow of the water which can be found from the environmental conditions such as wind and ocean current models (Petrich and Subbarao (2011)).

We assume that all robots move at a speed of $S_r > S_p$. We further assume that all robots can communicate with each other at all times and thus, restrict our attention to centralized algorithms.

We focus on the mapping problem in this paper. Therefore, we assume that all robots start at the same location where they first observe the ROI. We seek tours for each robot that explore the ROI and return back to this starting location.

Problem 1. Multi-Robot Exploration of Translating ROI. *Find a tour for all the robots that minimizes the exploration time such that every point in the ROI is visible from the sensor footprint of at least one robot's tour. All tours must return to the same starting position. The exploration time is given by the time when the last robot returns to the starting position.*

The proposed problem is an online exploration problem. The objective function is the exploration time which is the time of the longest tour. In the next section, we present an algorithm that is based on recursive DFS which is competitive with respect to the optimal solution.

4 ROI Exploration over a Grid Map

In this section, we present our main algorithm. We first solve a simpler version of Problem 1 where the ROI is approximated as a grid map. We then use this result to solve Problem 1 by relaxing the grid approximation afterwards. Our algorithm is based on the recursive DFS that models the ROI under exploration as a tree. We first show that our strategy is competitive for the grid map case and then analyze the effect of approximating an arbitrary ROI shape with a grid.

Recursive DFS Algorithm for a Grid Map

In this section, we assume that the ROI is represented as a grid map (Algfoor et al. (2015)). The environment is modeled as a collection of cells, each of which is a square of unit side length. Each cell is connected to four of its neighbors. The ROI P is just a collection of C cells that form one connected set (if a cell $c \in P$ is part of the ROI, then one of its four neighbors must also be a part of the ROI when $C > 1$).

The problem of exploring the ROI is then simplified to that of exploring a grid map and identify the cells that belong in P. Since we assume that the sensor footprint is also a unit square, a robot may obtain an image by positioning itself at the center of a cell. By analyzing the pixels on the boundary of the image, the robot can then determine if any of the four neighboring cells are also part of the ROI or not.

We model P as a tree and propose a recursive DFS algorithm based on the tree exploration algorithm given by Higashikawa et al. (2014). Higashikawa et al. (2014) developed a recursive DFS algorithm for exploring a binary tree. In our case, the grid graph to be explored is not necessarily a tree (it may contain cycles). Regardless, we show that modeling the underlying graph as a binary tree still leads to an algorithm with a constant competitive ratio.

The root of the tree is the cell corresponding to the starting position of the robots. Upon visiting a cell, the robots can identify if one or more of the four neighboring cells also contain the ROI. The neighboring cells that contain the ROI

[†]This is equivalent to the rigid-body translation of P.

are added as children of the present cell in the tree unless those cells have been previously added to the tree. This condition prevents cycles.

The number of neighboring cells when a robot visits a new cell can be at most three. Therefore, the resulting tree may not be binary. However, by introducing a dummy edge of length 0 and a dummy vertex, we can convert the tree into a binary tree without loss of generality.[‡]

Each neighboring ROI cell determined by the sensing model becomes one of candidate cells that robots can choose from as the next vertex to visit. The goal becomes to visit all $C-1$ cells (that correspond to the ROI cells but excluding the starting cell) at least once by one of the robots.

If $R = 1$, then our algorithm becomes conventional recursive DFS for a single robot. However, in the multi-robot case, as the robots build the tree, we split the robots as equally as possible and assign them to explore the children vertices.

We define three states for each vertex in the tree: *unexplored* if the vertex is not visited by any robots; *under exploration* if the vertex is visited by any robots but the leaf vertex connected from the vertex is not visited by any robots; and *explored* if the vertex as well as the leaf vertex in the same branch are visited by any robots. When robots decide which vertex to move among neighboring cells of an ROI region, they do not consider *explored* vertices but vertices that are either *unexplored* or *under exploration*. This is because having *explored* vertex means that the offspring of it must have also been explored by any robots (see Figure 4).

The details are given in Algorithm 1.[§] All vertices are marked as *unexplored* state in the beginning. Each robot runs Algorithm 1 whenever it reaches a vertex. The algorithm can be implemented to a single robot independently with respect to other robots as long as they can share the state information of vertices. The algorithm terminates when all robots return to the starting vertex and all vertices are marked as *explored*.

Theoretical Analysis

In this section we analyze the proposed Algorithm 1. We start with the upper bound analysis. We then show the lower bound for optimal algorithm, followed by the competitive analysis for the case of grid approximation.

Upper Bound Analysis To analyze the cost of the proposed algorithm, we adapt the reward[¶] collecting rule proposed by Higashikawa et al. (2014) to the case of a translating ROI. Note that this rule is not required for implementing the algorithm, but only for analyzing the competitive ratio.

Higashikawa et al. (2014) define the concept of a backbone and a rib in a tree (shown in Figure 4). The backbone is a path that starts from the root vertex and ends at one of the leaf vertices. The rib is a subtree generated by discarding the backbone and edges incident with the backbone from the original tree.

Let $l(e)$ be the length of an edge e. The length of an edge e is 0 if e is dummy edge or 1 otherwise. $L = \sum_{e \in E} l(e)$ be the sum of the total length of all edges in the tree. Note that $L = C - 1$ where the ROI consisting of C cells is represented by the tree structure.

(a) Two robots exploring the grid map.

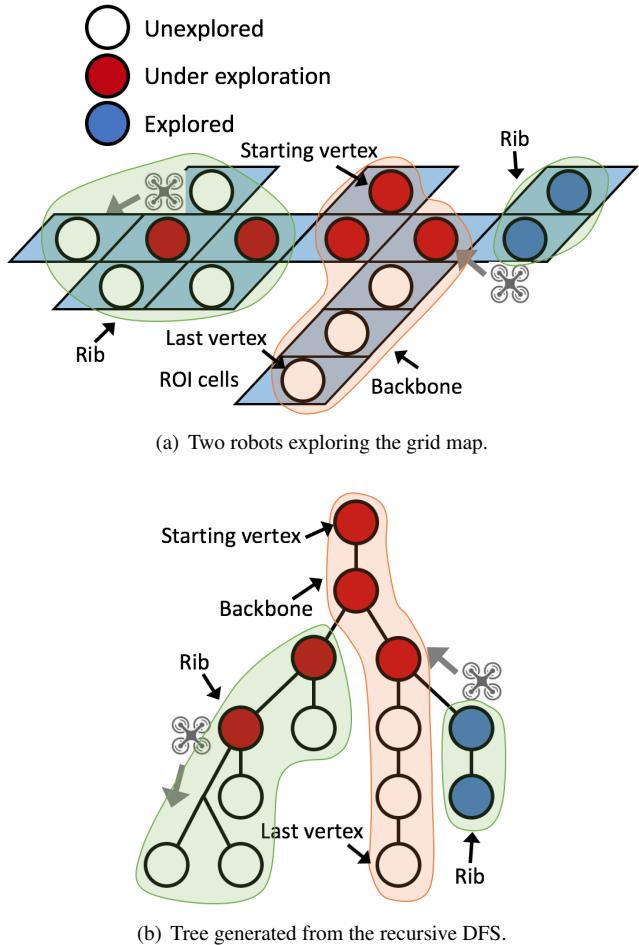(b) Tree generated from the recursive DFS.

Figure 4. Description of tree components. The binary tree consists of a backbone and a finite number of ribs. Each vertex is marked as one of *unexplored*, *under exploration* or *explored*.

Higashikawa et al. (2014) define two reward functions, each with a total reward of $l(e)$, on every rib edge e and $1 + \lfloor \log R \rfloor$ reward functions, again each with a total reward of $l(e)$, on every backbone edge e. The rewards are collected continuously by the robots following the rules described next: (1) Only one robot in a group traversing a rib edge in the forward direction for the first time collects a reward. (2) Only one robot in a group traversing a rib edge in the backward direction for the first time collects a reward. (3) Each of the $1 + \lfloor \log R \rfloor$ robots traversing a backbone edge in the forward direction for the first time collects a reward. (4) Only one robot in a group traversing a backbone edge after the first group collects a reward.

Let t_{last} be the time when the last robot reaches a leaf vertex in the tree. Higashikawa et al. (2014) show that the total sum of rewards collected by all the robots is at least $(1 + \lfloor \log R \rfloor)t_{last}$. This assumes that the robots move at unit speed and the tree is static. In our case, the tree (actually, ROI) is moving with a speed of S_p and the robots are moving

---

[‡]This step is included in Line 15 of Algorithm 1.

[§]In the algorithm, we use $N(v_i)$ to denote the neighborhood of the i-th vertex such that $N(v_i) = \{v_j \in V | (v_j, v_i) \in E\}$.

[¶]We use the term *reward function* to replace the term *token* defined in Higashikawa et al. (2014).

Algorithm 1: Multi-Robot Recursive DFS

1
2 Observe $N(v)$ to determine whether neighboring cells are ROI cells or non-ROI cells.
3 **if** $|N(v)|=0$ **then**
4   Mark v as *explored*.
5   Move back to the parent vertex ($\rightarrow$next vertex) and directly jump to Line 24.
6 **end**
7 Communicate with robots to update the state of $N(v)$, *i.e.*, *unexplored*, *under exploration*, and *explored*.
8 $N(v) \leftarrow N(v) \backslash \{explored$ vertices$\}$.
9 **if** $v' \in N(v)$ *is under exploration* **then**
10   **if** *moving to v' generates a cycle in the tree* **then**
11    $N(v) \leftarrow N(v) \backslash \{v'\}$.
12   **end**
13 **end**
14 **if** $|N(v)| > 1$ **then**
15   **if** $|N(v)| > 2$ **then**
16   Add a dummy edge of length 0 and a dummy vertex in order to keep the tree as a binary tree.
17   **end**
18   Split robots into two children as equally as possible.
19   Move to one of two children ($\rightarrow$next vertex) and mark v as *under exploration*.
20 **else if** $|N(v)| = 1$ **then**
21   Move to the child ($\rightarrow$next vertex) and mark v as *under exploration*.
22 **else if** $|N(v)| = 0$ **then**
23   Move back to the parent vertex ($\rightarrow$next vertex).
24 **end**
25 $v \leftarrow$ the next vertex.

with a speed of S_r. The reward collection rule does not change in this case. What changes is the total sum of rewards collected in t_{last} time. In our case, the total sum of rewards collected by all the robots will be at least $(S_r - S_p)(1 + \lfloor \log R \rfloor)t_{last}$. The term $(S_r - S_p)$ comes from the lower bound on the relative speeds of the robots and the ROI.

Higashikawa et al. (2014) also show that the total possible reward that the robots can collect is at most $2(L - d_{max}) + (1 + \lfloor \log R \rfloor)d_{max}$. Here d_{max} denotes the distance of the farthest vertex in the tree from the root. Therefore, we have:

$$(S_r - S_p)(1 + \lfloor \log R \rfloor)t_{last} \\ \leq 2(L - d_{max}) + (1 + \lfloor \log R \rfloor)d_{max}. \quad (1)$$

We denote the time taken by the proposed algorithm by ALG. We are now ready to state the upper bound on ALG.

Lemma 1. Upper Bound for Multi-Robot Recursive DFS.

$$\text{ALG} \leq \frac{2(C + d_{max}\lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \quad (2)$$

Proof. ALG can be upper bounded as follows:

$$\text{ALG} \leq t_{last} + \frac{d_{max}}{S_r - S_p}, \quad (3)$$

where $\frac{d_{max}}{S_r - S_p}$ is the time taken to traverse the longest length of the backbone when the robot and the ROI move away from

each other. By using Equation (1), we have:

$$t_{last} + \frac{d_{max}}{S_r - S_p} \leq \frac{2L + (\lfloor \log R \rfloor - 1)d_{max}}{(S_r - S_p)(1 + \lfloor \log R \rfloor)} + \frac{d_{max}}{S_r - S_p}, \quad (4)$$

$$= \frac{2(C - 1 + d_{max}\lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}, \quad (5)$$

$$\leq \frac{2(C + d_{max}\lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \quad (6)$$

Equation (5) is obtained by the fact that $L = C - 1$. Removing a negative term from Equation (5) completes the proof as Equation (6).

Corollary 1. Special Cases. *Upper bounds for the following special cases can be derived from Lemma 1, such as Multi-Robot Static ROI (MRSR), Single Robot Translating ROI (SRTR), and Single Robot Static ROI (SRSR).*

MRSR	$\text{ALG} \leq \frac{2(C + d_{max}\lfloor \log R \rfloor)}{S_r(1 + \lfloor \log R \rfloor)}$
SRTR	$\text{ALG} \leq \frac{2C}{S_r - S_p}$
SRSR	$\text{ALG} \leq \frac{2C}{S_r}$

Table 1. Upper bounds of special cases.

Note that the upper bound for MRSR becomes the result from Higashikawa et al. (2014) if $S_r = 1$. Also, the upper bound for SRSR is equivalent to Icking et al. (2000) if $S_r = 1$.

Proof. Please refer to Appendix 8.

Lower Bound Analysis We study the lower bound for the optimal algorithm in order to obtain a competitive ratio. Let OPT_g^1 be the time taken by the optimal algorithm to explore a grid map when using a single robot. The lower bound can be constructed as:

$$\text{OPT}_g^1 \geq \frac{C - 1}{S_r + S_p}. \quad (7)$$

We use OPT_g^R to represent the time taken by the optimal algorithm over any grid polygon of an ROI region using R robots. Then, the following lemma gives the lower bound for OPT_g^R.

Lemma 2. Lower Bound for Optimal Algorithm.

$$OPT_g^R \geq \frac{C - 1}{(S_r + S_p)R}. \quad (8)$$

Proof. Please refer to Appendix 8.

Theorem 1. Competitive Ratio over the Grid Polygon. *The competitive ratio of Algorithm 1 for a grid map is:*

$$\text{ALG} \leq \frac{2(S_r + S_p)(R + \lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)} OPT_g^R \\ + \frac{2}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \quad (9)$$

Proof. Substituting Equation (8) into Equation (2) gives:

$$\text{ALG} \leq \frac{2((S_r + S_p)R\text{OPT}_g^R + 1 + d_{max}\lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \quad (10)$$

Since $\frac{d_{max}}{S_r + S_p} \leq \text{OPT}_g^R$, it follows:

$$\text{ALG} \leq \frac{2(S_r + S_p)(R + \lfloor \log R \rfloor)\text{OPT}_g^R + 2}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \quad (11)$$

5 ROI Exploration over an Arbitrary ROI Shape

The presented results so far are for a grid map approximation of the ROI. In this section, we will relate the bounds obtained for the grid map case to the case of arbitrarily shaped ROIs. Specifically, we will extend Lemma 1 to apply to an ROI region that may have an arbitrary shape.

The algorithm for exploring the ROI remains the same. We will still construct a tree that represents a grid map of the ROI. The main difference here is that in the previous analysis, we assumed that the boundary of the ROI matched the boundary of a grid map exactly. This will no longer hold. Instead, we will explore a grid map that is an *outer* approximation of the ROI (Figure 5).

We define C_{out}^{ALG} and C_{in}^{ALG} to denote the number of cells in the outer and inner grid approximation by our algorithm, respectively. The outer grid map completely contains the ROI whereas the inner grid map lies completely inside the ROI. Therefore, the term C in the upper bound (Lemma 1) will now be replaced by C_{out}^{ALG}. However, the C term in the lower bound (Lemma 2) cannot be replaced by C_{in}^{ALG}. This is because C_{in}^{ALG} is defined by the grid imposed by our algorithm. It may be possible to have another grid map (of the same unit side length) that is oriented and/or translated such that it contains fewer than C_{in}^{ALG} cells in the interior. We will first find the relationship between C_{out}^{ALG} and C_{in}^{ALG}. Then, we will relate C_{in}^{ALG} to C_{in}^{BEST} which is the best grid that contains the fewest number of cells completely inside the ROI.

By a slight abuse of notation, we interchangeably use C_{out}^{ALG} and C_{in}^{ALG} to also denote the corresponding set of cells (along with denoting the number of cells in the set).

Lemma 3. Grid Approximation of Arbitrary ROI Shape. *The upper bound on C_{out}^{ALG} for a* fat *polygon (from Definition 1) is given by:*
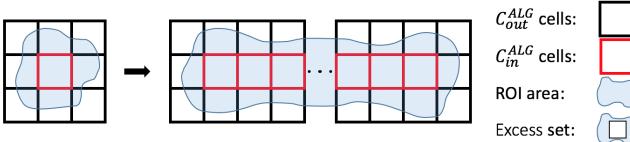
$$C_{out}^{ALG} \leq 3C_{in}^{ALG} + 6. \quad (12)$$

Figure 5. Row formation of C_{in}^{ALG} cells as the number of cells changes from 1 to a finite number.

Proof. To prove the lemma, we define an EXCESS set that contains all cells, $p \in P \backslash C_{in}^{ALG}$. That is, EXCESS set contains all cells in C_{out}^{ALG} but not in C_{in}^{ALG}. Therefore, the size of the EXCESS set is equal to $C_{out}^{ALG} - C_{in}^{ALG}$. We prove the lemma in three steps.

EXCESS is maximum if all cells in C_{in}^{ALG} form a convex polygon. If there is a reflex vertex$^{\|}$ in C_{in}^{ALG}, then the reflex vertex cell does not contribute any cell to the EXCESS set that is already contributed by one of the neighbors of the reflex vertex. Since C_{in}^{ALG} is a grid map, the only convex shape possible is a rectangle.

If the width of the rectangle is equal to one, then each cell in C_{in}^{ALG} contributes two cells that are in the EXCESS set (one above and one below) in addition to three more cells on either end point. This is shown in Figure 5. Therefore, the size of EXCESS set is equal to $2C_{in}^{ALG} + 6$.

If the width of the rectangle is more than 1, then each cell will contribute at most one addition cell in the EXCESS set. Therefore, the size of the EXCESS set is less than or equal to $C_{in}^{ALG} + 8$.

The width of the rectangle cannot be less than 1; it violates the *fat* condition for the polygon. Therefore, the maximum possible value for the size of the EXCESS set is $2C_{in}^{ALG} + 6$. By substituting EXCESS with $C_{out}^{ALG} - C_{in}^{ALG}$, we have Equation (12).

The grid corresponding to C_{out}^{ALG} and C_{in}^{ALG} is generated by the proposed algorithm. It is possible that there exists some other grid which has fewer than C_{in}^{ALG} cells completely contained within the ROI. It may not be possible to generate this "best" grid due to the nature of online exploration. Nevertheless, we analyze how the relationship between C_{in}^{ALG} and C_{in}^{BEST}. We define C_{in}^{BEST} to denote the fewest number of cells in the inner grid approximation that is completely contained in the ROI (and adding any other cell to C_{in}^{BEST} would not allow C_{in}^{BEST} to be completely inside the ROI). The relationship is given by:

Lemma 4. Best Possible Grid-Approximation.

$$C_{in}^{ALG} \leq 6C_{in}^{BEST}. \quad (13)$$

Proof. To prove this relationship, it is sufficient to consider any grid approximation (generated by any algorithm) with respect to the best grid approximation.
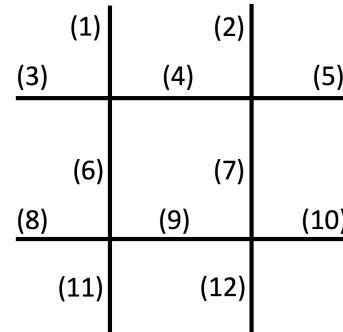
Figure 6. A part of grid cell from any grid approximation. Unique number is assigned to a different side of grid cells.

Figure 6 shows a part of grid cells generated by any grid approximation. Each number in the figure corresponds to a

---

$^{\|}$A vertex is called the reflex vertex if the external angle formed by two edges incident to this vertex is less than $180°$.

different side of grid cells. Let C_{in}^{BEST} be a single grid cell generated from the best grid approximation that overlaps with the central cell $(4, 6, 7, 9)$ without loss of generality. Our observation is that the number of crossings is equal to the number of cells in C_{in}^{ALG} that C_{in}^{BEST} overlaps.

We prove that 7 crossings are impossible. In order to cross more than four edges, C_{in}^{BEST} has to cross all of the $(4, 6, 7, 9)$ edges. In addition, it must cross three of $(1, 2, 3, 5, 8, 10, 11, 12)$ edges. Let us consider the case when edge (1) is crossed. The other cases are symmetric. If edge (1) is crossed, then crossing $(5, 8, 10, 11, 12)$ is impossible since these edges are more than a unit distance apart. Only (2) and (3) edges are only available edges to cross more. However, if C_{in}^{BEST} crosses both (2) and (3) edges, the side length of C_{in}^{BEST} becomes greater than 1. Therefore, C_{in}^{BEST} cannot cross more than seven edges. Therefore, $C_{in}^{ALG} \leq 6 C_{in}^{BEST}$.

Finally we give our main result as follows:

Theorem 2. Competitive Ratio for Arbitrary ROI Shape. *Let OPT^R be the time taken by the optimal algorithm over any arbitrary ROI shape using R robots.*

$$\begin{aligned} ALG \leq & \frac{2(S_r + S_p)(18R + \lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)} OPT^R \\ & + \frac{48}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \end{aligned} \tag{14}$$

Proof. Although OPT^R is the cost for any arbitrary ROI shape, we can still lower bound this using C_{in}^{BEST} (similar to Lemma 2) as:

$$\mathrm{OPT}^R \geq \frac{C_{in}^{BEST} - 1}{(S_r + S_p)R}. \tag{15}$$

Let $(S_r - S_p)(1 + \lfloor \log R \rfloor)$ be $\mathcal{M}$. We can obtain the following inequalities from Lemmas 1, 3, and 4 as follows.

$$\mathrm{ALG} \leq \alpha C_{out}^{ALG} + a \leq \beta C_{in}^{ALG} + b \leq \gamma C_{in}^{BEST} + b, \tag{16}$$

where $\alpha = \frac{2}{\mathcal{M}}$, $a = \frac{2 d_{max} \lfloor \log R \rfloor}{\mathcal{M}}$, $\beta = \frac{6}{\mathcal{M}}$, $b = \frac{12 + 2 d_{max} \lfloor \log R \rfloor}{\mathcal{M}}$, and $\gamma = \frac{36}{\mathcal{M}}$.

Substituting Equation (15) into the last inequality of Equation (16) and using $\frac{d_{max}}{S_r + S_p} \leq \mathrm{OPT}^R$, we have:

$$\begin{aligned} \mathrm{ALG} & \leq \frac{36(S_r + S_p)R}{\mathcal{M}} \mathrm{OPT}^R + \frac{48 + 2 d_{max} \lfloor \log R \rfloor}{\mathcal{M}}, \\ & \leq \frac{2(S_r + S_p)(18R + \lfloor \log R \rfloor)}{\mathcal{M}} \mathrm{OPT}^R + \frac{48}{\mathcal{M}}. \end{aligned} \tag{17}$$

6 Simulations

We empirically evaluated our algorithm using MATLAB simulations. Specifically, we verified the performance of the proposed recursive DFS for the grid map approximation of the ROI (Theorem 1).

We randomly generated a set of ROI grid maps. We randomly chose one of four directions (*i.e.*, north, south, east, and west) for the direction of translation of the ROI. The assumption that the moving direction of the ROI is known a priori enables robots to align the axis of the grid map with that direction although robots still do not know about best

approximation for grid map. Figure 7 (a) shows an example of the generated ROI that consists of 200 cells.

We measured the cost of our algorithm as well as the upper and lower bounds by changing the number of ROI cells, the number of robots, and the speed ratio between the robot and the ROI. The lower bound in our analysis is given in Lemma 2. In deriving the lower bound, we assume that the robots always travel opposite to the ROI, thereby yielding the lowest possible time. In practice, the robots will not always travel in this best possible direction. Therefore, we find another lower bound using a baseline lawn mower algorithm. We assume that this baseline algorithm knows the tightest rectangle that is guaranteed to contain the ROI. The axis of the rectangle is aligned with the direction of translation of the ROI. Given R robots, we split this rectangle into R smaller ones. We can split this rectangle either along its length or its breadth. The exploration time will be different in each case. We find the time required to cover each smaller rectangle using a lawn mower strategy in both cases and take the lower one. We also ignore the time required for the robots to go from the starting position to the smaller rectangles. This is clearly a lower bound for any online strategy that does not know the size of the ROI. Nevertheless, we find that the exploration time for our algorithm is comparable to this lower bound.

Each case was obtained from 100 randomly generated trials (see Figures 7 (b–d)). Figure 7 (b) shows that the expected exploration time for all cases is proportional to the number of cells in the grid ROI. The difference between the maximum and minimum costs as well as that between the upper bound and the cost of the algorithm also becomes larger as more ROI cells are to be explored. The cost of our algorithm is closer to the lower bound than the upper bound which becomes even more pronounced as the number of cells increases.

Figure 7 (c) plots the exploration time when changing the number of robots. Not surprisingly, the exploration time of our algorithm and the lower bound decrease as the number of robots increases. The upper bound depends on d_{max}. As a result, the upper bound does not decrease as much as the other curves since d_{max} can be high for randomly generated maps. Regardless, we find that our algorithm performs better empirically than what is given by the theoretical upper bound.

Figure 7 (d) shows the exploration time when changing the speed ratio between the robot and the ROI, *i.e.*, $\frac{S_r}{S_p}$. The exploration time for our algorithm and the upper bound decrease as the speed ratio increases. We also observe that the difference between upper and lower bounds decreases as the speed ratio increases.

The simulation results verify the theoretical upper and lower bounds determined by our analysis. In addition, they demonstrate that the practical performance of our algorithm is better than that indicated by the upper bounds. We observe that the practical performance is closer to the lower bound than the upper bound.

7 Field Experiment

In this section, we conduct proof-of-concept experiments using a single UAV equipped with a downward-facing
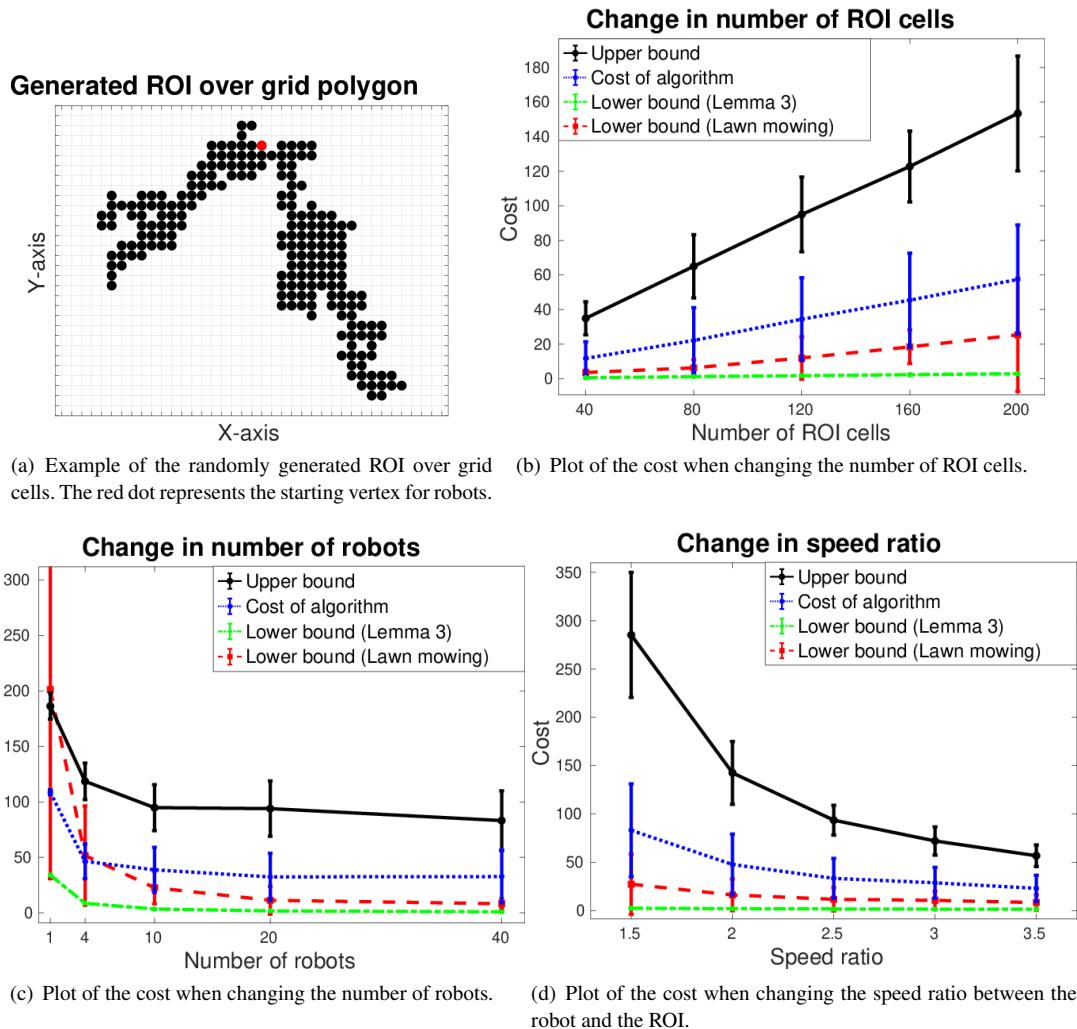
Generated ROI over grid polygon

(a) Example of the randomly generated ROI over grid cells. The red dot represents the starting vertex for robots.

Change in number of ROI cells

(b) Plot of the cost when changing the number of ROI cells.

Change in number of robots

(c) Plot of the cost when changing the number of robots.

Change in speed ratio

(d) Plot of the cost when changing the speed ratio between the robot and the ROI.

Figure 7. Simulation results. We fixed the number of ROI cells, the number of robots, the speed ratio as $120, 20, 2.5$, respectively, when the corresponding variable was not a subject to be changed. We ran 100 trials for each case. Each case is plotted as mean, maximum and minimum values from 100 trials.

camera to monitor a stationary region of unknown size and shape. In a practical implementation, there are a number of design choices that must be made (*e.g.*, what altitude to fly? how to convert the camera images into cell measurements? how to deal with erroneous sensor measurements?). In this section, we answer these questions in the context of our system.
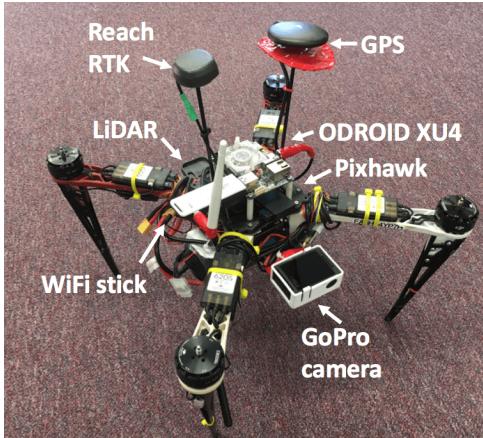
Our environment to be mapped is a $92m \times 21m$ long runway which serves as a proxy of the ROI. Figure 8 shows hardware details of the UAV and the snapshot of the environment. The UAV has ODROID-XU4 single-board computer which runs Ubuntu 16.04 with ROS Kinetic (Quigley et al. (2009)). The onboard software controls the UAV, communicates with GoPro HERO4 camera over WiFi to read sensor information, and detects the runway.

Our planning strategy consists of two modes: (1) lawn mowing and (2) the single-robot version of Algorithm 1. The input to the lawn-mowing mode is a bounding box that is guaranteed to contain at least some part of the runway. In the lawn-mowing mode, the UAV sweeps the bounding box. As soon as the UAV observes a part of the runway, the UAV switches to the recursive DFS algorithm.

Once in the recursive DFS mode, we discretize the environment into grid cells. The grid is aligned with the UTM coordinates (Wikipedia contributors (2019)). The origin of the grid is placed at the starting location of the UAV. Note that the grid is not aligned with the runway (Figure 8 (b)) and the location, shape, and size of the runway are not given to the UAV a priori.

Each cell is of size $4m \times 4m$. We use the onboard GoPro images to determine if a cell corresponds to the runway. Each image is divided into 3×3 regions (Figure 9). The size of a grid map cell ($4m \times 4m$) corresponds to the footprint of the center region in the image when flying at an altitude of $12m$. The center and the top, left, bottom, and right regions in the image (refer Figure 9) are used to determine if the current cell and its four neighbors in the grid map contain the runway or not.

Each image is first converted into grayscale and thresholded (at the intensity value of 150). Then, the thresholded image is dilated (7 times) so that the gaps in the grass region are filled in (refer the left image in Figure 9). Consequently, the entire grass region becomes filled with white pixels and the runway region mostly comprises black pixels. Then the percentage of black pixels

(a) UAV platform.

(b) Runway ($92m \times 21m$) to be explored.

Figure 8. Experimental setting.

in the individual regions is calculated, for each highlighted neighbouring region in the figure. Finally, we produce a binary classification result based on if this percentage is above or below a threshold.

Figure 9. Example of our sensing model using a single image that contains both the runway and the grass region. The left image is the thresholded image. The right image shows the detection result indicating the percentage of black pixel values printed on the grid cells (colored in red in four neighboring cells).

Our classifier may not give a correct detection result due to a number of reasons. First, it may produce false positives and false negatives. Second, the detection result is sensitive to the intensity threshold value we set. Third, even if the UAV is on top of the current cell, the camera may point to a wrong cell due to pitch and roll used to counter wind disturbance and imperfect flight controller. Lastly, the change in the sunlight condition might produce noisy measurements. Therefore, instead of relying on a single image, we use five images per cell. If three of these images mark a cell as containing the runway, we treat it as a positive detection. When mapping a region, we care more about the completeness of exploration rather than the efficiency. Therefore, we make the following conservative design choices. If a cell is marked as containing the runway, it will never be reversed later on, even if a

future measurement taken from some other location yields the opposite detection. On the other hand, if a cell is marked as not containing the runway and a subsequent measurement suggests otherwise, we will mark it as a positive runway detection.

The measured flight time of the UAV with a single battery is approximately 12 minutes, which is not enough for finishing the exploration task. Therefore, we designed our software in a way to keep track of its previous computation even after replacing the battery. To do that, the software stores the information of the generated tree (*i.e.*, the status of vertices and the tree structure) and feeds that to the next flight in order for the UAV to start from the vertex visited last in the previous flight.

Figure 10 shows the result of the runway exploration experiment. We flew the UAV at a nominal speed of $1m/s$ and the ambient wind speed was approximately $3.6m/s$. The total flight time was 1 hour and 13 minutes and consisted of six battery replacements. The final tree contained 143 vertices.

The final grid map overlaid on Google Earth is shown in Figure 11. The percentage of the area of intersection between the ground-truth and the final map, normalized by the area of their union, was 75.7%.

We also compute the false-positive and false-negative detection rates. Out of the total 483 detections, 27 were false positives and 53 were false negatives. Note that all but two cells that gave false-negative detections, eventually gave a positive detection. As a result, the final map (Figure 11) has only two cells that are incorrectly mapped as not being part of the runway. The cells that are just outside the boundary, however, are incorrectly mapped as being part of the runway. This is likely because of our conservative exploration choice of marking a cell as containing the runway even if a single detection is positive.

In summary, we present how our algorithm handles real-world issues and can be implemented on actual robots. We show the efficacy of the proposed scheme through field experiments.

8 Conclusion

We propose a recursive DFS algorithm for a team of aerial robots to explore a translating ROI without knowing its size and shape. We present two approaches for the given problem where the first approximates the ROI to map to the grid whereas the second considers any arbitrary shape of ROI as long as it is *fat*. Both approaches are competitive with respect to the optimal algorithm. We demonstrate the performance of our algorithm through proof-of-concept deployment to map a stationary ROI.

One of the practical concerns not modeled by our system is that robots, especially UAVs, have limited battery lifetime. As such, we would like to devise algorithms that can map the ROI subject to the limited battery lifetime constraint. In particular, in our formulation, we restrict the UAV to fly at a fixed altitude. However, one may be able to extend the coverage range by flying at higher altitudes. An interesting and relevant extension of this work would be to plan in 3D space as opposed to just 2D.
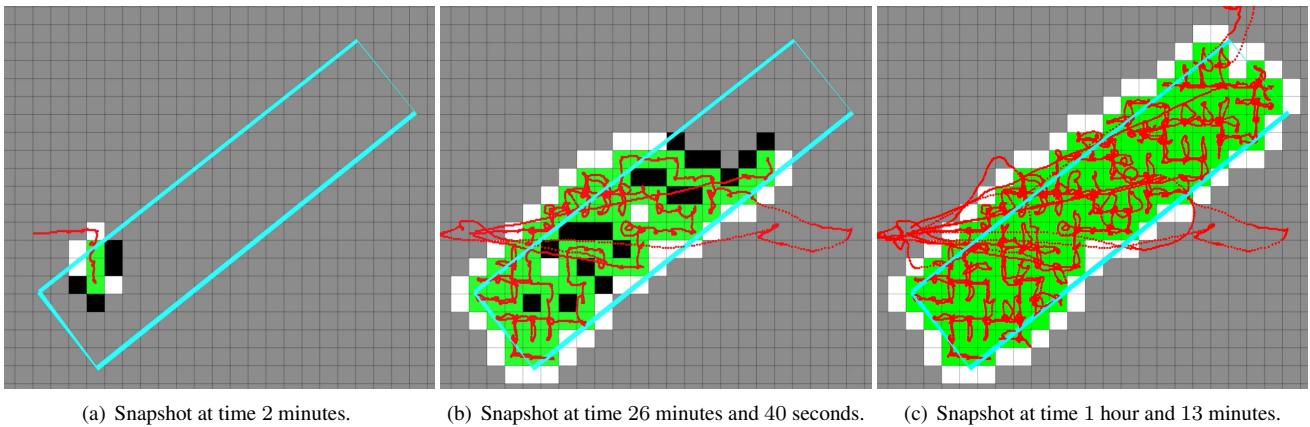
(a) Snapshot at time 2 minutes. (b) Snapshot at time 26 minutes and 40 seconds. (c) Snapshot at time 1 hour and 13 minutes.

Figure 10. Experimental result. The blue square line represents the ground truth of the runway to be explored that is unknown initially. The red line denotes the trajectory of the UAV. The size of each cell is $4m \times 4m$ and gray, white, black and green colors represent unknown, non-runway, unexplored runway and explored cells, respectively. The total flight time taken to completely explore the entire runway was 1 hour and 13 minutes, consisting of six battery replacements. The video is available at: `https://youtu.be/ZTpPb0C4Lk0`.
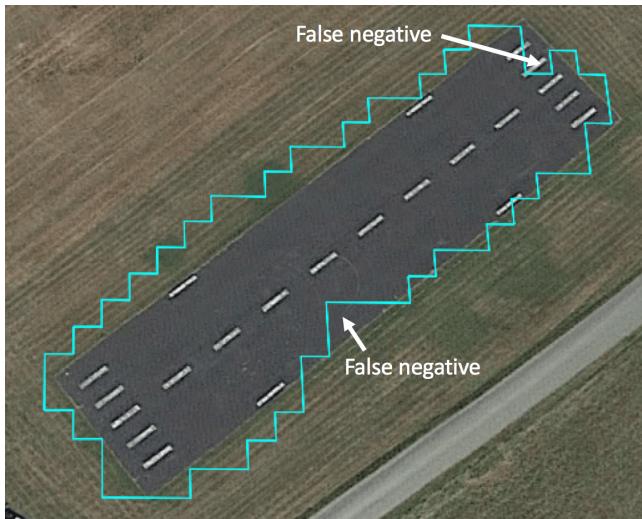

Figure 11. Resultant boundary of the runway region (*i.e.*, the boundary of the grid map) plotted on Google Earth colored in blue.

Acknowledgements

References

Algfoor ZA, Sunar MS and Kolivand H (2015) A comprehensive study on pathfinding techniques for robotics and video games. *International Journal of Computer Games Technology* 2015: 7.

Aloupis G, Bose P, Dujmović V, Gray C, Langerman S and Speckmann B (2014) Triangulating and guarding realistic polygons. *Computational geometry* 47(2): 296–306.

Arkin EM, Fekete SP and Mitchell JS (2000) Approximation algorithms for lawn mowing and milling. *Computational Geometry* 17(1-2): 25–50.

Arya S, Cheng SW and Mount DM (2001) Approximation algorithm for multiple-tool milling. *International Journal of Computational Geometry & Applications* 11(03): 339–372.

Bender MA, Fernández A, Ron D, Sahai A and Vadhan S (2002) The power of a pebble: Exploring and mapping directed graphs. *Information and computation* 176(1): 1–21.

Borodin A and El-Yaniv R (2005) *Online computation and competitive analysis*. cambridge university press.

Brass P, Cabrera-Mora F, Gasparri A and Xiao J (2011) Multirobot tree and graph exploration. *IEEE Transactions on Robotics* 27(4): 707–717.

Cesare K, Skeele R, Yoo SH, Zhang Y and Hollinger G (2015) Multi-uav exploration with limited communication and battery. In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, pp. 2230–2235.

Corah M and Michael N (2019) Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice. *Autonomous Robots* 43(2): 485–501.

Das J, Cross G, Qu C, Makineni A, Tokekar P, Mulgaonkar Y and Kumar V (2015a) Devices, systems, and methods for automated monitoring enabling precision agriculture. In: *Automation Science and Engineering (CASE), 2015 IEEE International Conference on*. IEEE, pp. 462–469.

Das S, Dereniowski D and Karousatou C (2015b) Collaborative exploration by energy-constrained mobile robots. In: *International Colloquium on Structural Information and Communication Complexity*. Springer, pp. 357–369.

Das S, Flocchini P, Kutten S, Nayak A and Santoro N (2007) Map construction of unknown graphs by multiple agents. *Theoretical Computer Science* 385(1-3): 34–48.

Dunbabin M and Marques L (2012) Robots for environmental monitoring: Significant advancements and applications. *IEEE Robotics & Automation Magazine* 19(1): 24–39.

Dynia M, Łopuszański J and Schindelhauer C (2007) Why robots need maps. In: *International Colloquium on Structural Information and Communication Complexity*. Springer, pp. 41–50.

Efrat A (2005) The complexity of the union of (α,β)-covered objects. *SIAM Journal on Computing* 34(4): 775–787.

Fahad M, Guo Y, Bingham B, Krasnosky K, Fitzpatrick L and Sanabria FA (2017) Robotic experiments to evaluate ocean plume characteristics and structure. In: *Intelligent Robots and*

Systems (IROS), 2017 IEEE/RSJ International Conference on.* IEEE, pp. 6098–6104.

Fernandez Galarreta J, Kerle N and Gerke M (2015) Uav-based urban structural damage assessment using object-based image analysis and semantic reasoning. *Natural hazards and earth system sciences* 15(6): 1087–1101.

Fraigniaud P, Gasieniec L, Kowalski DR and Pelc A (2006) Collective tree exploration. *Networks: An International Journal* 48(3): 166–177.

Galceran E and Carreras M (2013) A survey on coverage path planning for robotics. *Robotics and Autonomous systems* 61(12): 1258–1276.

Girdhar Y, Giguere P and Dudek G (2014) Autonomous adaptive exploration using realtime online spatiotemporal topic modeling. *The International Journal of Robotics Research* 33(4): 645–657.

Higashikawa Y, Katoh N, Langerman S and Tanigawa Si (2014) Online graph exploration algorithms for cycles and trees by multiple searchers. *Journal of Combinatorial Optimization* 28(2): 480–495.

Hitz G, Gotovos A, Garneau MÉ, Pradalier C, Krause A, Siegwart RY et al. (2014) Fully autonomous focused exploration for robotic environmental monitoring. In: *2014 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, pp. 2658–2664.

Icking C, Kamphans T, Klein R and Langetepe E (2000) Exploring an unknown cellular environment. In: *EuroCG.* pp. 140–143.

Icking C, Kamphans T, Klein R and Langetepe E (2005) Exploring simple grid polygons. In: *International Computing and Combinatorics Conference.* Springer, pp. 524–533.

Ishida H, Wada Y and Matsukura H (2012) Chemical sensing in robotic applications: A review. *IEEE Sensors Journal* 12(11): 3163–3173.

Juliá M, Gil A and Reinoso O (2012) A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots* 33(4): 427–444.

Kolenderska A, Kosowski A, Małafiejski M and Żyliński P (2009) An improved strategy for exploring a grid polygon. In: *International Colloquium on Structural Information and Communication Complexity.* Springer, pp. 222–236.

Lee SK, Fekete SP and McLurkin J (2016) Structured triangulation in multi-robot systems: Coverage, patrolling, voronoi partitions, and geodesic centers. *The International Journal of Robotics Research* 35(10): 1234–1260.

Lochmatter T, Göl EA, Navarro I and Martinoli A (2013) A plume tracking algorithm based on crosswind formations. In: *Distributed Autonomous Robotic Systems.* Springer, pp. 91–102.

Mahadev A, Krupke D, Fekete SP and Becker AT (2017) Mapping and coverage with a particle swarm controlled by uniform inputs. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE, pp. 1097–1104.

Megow N, Mehlhorn K and Schweitzer P (2012) Online graph exploration: New results on old and new algorithms. *Theoretical Computer Science* 463: 62–72.

Nuske S, Choudhury S, Jain S, Chambers A, Yoder L, Scherer S, Chamberlain L, Cover H and Singh S (2015) Autonomous exploration and motion planning for an unmanned aerial vehicle navigating rivers. *Journal of Field Robotics* 32(8): 1141–1162.

Petrich J and Subbarao K (2011) On-board wind speed estimation for uavs. In: *AIAA Guidance, Navigation, and Control Conference.* p. 6223.

Plonski PA, Vander Hook J, Peng C, Noori N and Isler V (2017) Environment exploration in sensing automation for habitat monitoring. *IEEE Transactions on Automation Science and Engineering* 14(1): 25–38.

Preshant A, Yu K and Tokekar P (2016) A geometric approach for multi-robot exploration in orthogonal polygons. In: *Workshop on Algorithmic Foundations of Robotics (WAFR).* URL http://www.wafr.org/papers/WAFR_2016_paper_25.pdf.

Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, Wheeler R and Ng AY (2009) Ros: an open-source robot operating system. In: *ICRA workshop on open source software*, volume 3. Kobe, Japan, p. 5.

Sim R and Little JJ (2009) Autonomous vision-based robotic exploration and mapping using hybrid maps and particle filters. *Image and Vision Computing* 27(1-2): 167–177.

Singh A, Krause A, Guestrin C and Kaiser WJ (2009) Efficient informative sensing using multiple robots. *Journal of Artificial Intelligence Research* 34: 707–755.

Sung Y and Tokekar P (2019) A competitive algorithm for online multi-robot exploration of a translating plume. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).* URL https://arxiv.org/abs/1811.02769v1. To appear.

Tokekar P, Bhadauria D, Studenski A and Isler V (2010) A robotic system for monitoring carp in minnesota lakes. *Journal of Field Robotics* 27(6): 779–789.

Tokekar P, Branson E, Vander Hook J and Isler V (2013) Tracking aquatic invaders: Autonomous robots for monitoring invasive fish. *IEEE Robotics & Automation Magazine* 20(3): 33–41.

Tokekar P, Vander Hook J, Mulla D and Isler V (2016) Sensor planning for a symbiotic uav and ugv system for precision agriculture. *IEEE Transactions on Robotics* 32(6): 1498–1511.

van der Stappen AF and Overmars MH (1994) Motion planning amidst fat obstacles. In: *Proceedings of the tenth annual symposium on Computational geometry.* ACM, pp. 31–40.

Wikipedia contributors (2019) Universal transverse mercator coordinate system — Wikipedia, the free encyclopedia. URL https://en.wikipedia.org/w/index.php?title=Universal_Transverse_Mercator_coordinate_system&oldid=891899172. [Online; accessed 29-April-2019].

Proof of Corollaries

The upper bound for MRSR can simply be obtained by plugging $S_p = 0$ into Equation (2) of Lemma 1.

The upper bound for SRTR can be derived from the upper bound of MRSR by having $R = 0$. However, we can even tighten the bound by using the following observation: if the robot and the ROI move toward each other in one direction, they must move away from each other in order to return to the starting location, and vice versa. Therefore, ALG can be upper bounded as:

$$\text{ALG} \leq \frac{C-1}{S_r + S_p} + \frac{C-1}{S_r - S_p}. \tag{18}$$

Taking out negative terms from the above equation becomes:

$$\mathrm{ALG} \leq \frac{2 S_r C}{(S_r + S_p)(S_r - S_p)}, \qquad (19)$$

which is a tighter bound than $\frac{2C}{S_r - S_p}$. Note that the difference between these bounds is $\frac{S_r}{S_r + S_p}$ that satisfies $\frac{1}{2} < \frac{S_r}{S_r + S_p} \leq 1$ because $S_r > S_p$.

The upper bound for SRSR can be derived by plugging either $R = 1$ and $S_p = 0$ into the upper bound for MRSR or $S_p = 0$ into the upper bound for SRTR.

Proof of Lemma 2

We claim the following inequalities.

$$\mathrm{OPT}^R \leq \mathrm{OPT}^1, \qquad (20)$$

This can be obtained from the fact that the more number of robots are deployed, the shorter time will be taken to explore the entire tree.

Consider a tree consisting of R branches. Then, we claim the following inequality:

$$\mathrm{OPT}^1 \leq R \mathrm{OPT}^R, \qquad (21)$$

Since OPT^R is the time for a robot to explore the longest branch in the tree, $R \mathrm{OPT}^R$ must be no less than OPT^1.

Combining these inequalities and Equation (7), we prove Lemma 2.