A Competitive Algorithm for Online Multi-Robot Exploration of a Translating Plume

Yoonchang Sung, *Student Member, IEEE,* Deeksha Dixit, and Pratap Tokekar, *Member, IEEE*

***Abstract*—In this paper, we study the problem of exploring a translating plume with a team of aerial robots. The shape and the size of the plume are unknown to the robots. The objective is to find a tour for each robot such that they collectively explore the plume. Specifically, the tours must be such that each point in the plume must be visible from the field-of-view of some robot along its tour. We propose a recursive Depth-First Search (DFS)-based algorithm that yields a constant competitive ratio for the exploration problem. 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 plume speed, respectively. We also consider a more realistic scenario where the plume 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 the *fat* condition. We empirically verify our algorithm using simulations as well as a proof-of-concept experiment mapping a stationary region.**

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

I. Introduction

We investigate the problem of exploring and mapping flows of an unknown hazardous agent in aquatic environments using a team of autonomous aerial robots. Our overall vision is to develop algorithms for enabling a team of robots to assist emergency responders in disaster scenarios, such as dispersal of oil aerosols and radioactive particulates in the environment. Previous work has shown the value of using Unmanned Surface Vehicles (USVs) for monitoring and sampling spatiotemporal plumes in aquatic environments [1], [2]. However, USVs can only provide a narrow (local) view of the plumes. Detecting a hazardous agent in the environment may require a USV to cover a large portion of the aquatic system, which may take a considerable amount of time. Furthermore, even after detecting the threat, teams of USVs may not be able to keep up with the rapidly spreading plume. Thus, emergency responders may not be informed of the full extent of the hazards, limiting their ability to respond quickly and effectively. This motivates the use of Unmanned Aerial Vehicles (UAVs) which can provide a wider (regional) picture and that can coordinate with USVs for more targeted deployments.

Teams of UAVs can collectively track the plumes and act as scouts to direct the USVs to sense for hazardous regions of interest (Figure 1). As a first step towards enabling coordination between UAVs and USVs, in this paper, we focus on the problem of mapping the extent of a 2D plume.

The problem of exploring an unknown 2D environment is a well-studied one in the robotics [3]–[6] and computational geometry [7]–[10] communities. The problem considered in this paper differs from these works in two critical ways. First, we consider the case where the plume is not static but is instead translating with a given velocity. As a result, the performance of the algorithm depends on the relative speeds of the robots and the plume. Second, in our setup the robots are not restricted to stay inside (or over) the plume all the time. The robots can fly over locations that are not part of the plume, thereby allowing them to "shortcut" from one part of the plume to the other. 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 plume a priori, they may not be able to take a "shortcut" even if one exists. As a result, the robots may ending up taking a longer path, resulting in a poorer performance. Nevertheless, we present an algorithm that is *competitive* with respect to the optimal algorithm.

We use the notion of *competitive ratio* [11] 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 plume 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 plume for a fixed number of robots. The constant depends on the relative speeds of the plume and the robots.

We require the robots to ensure that all points of the plume are within the Field-Of-View (FOV) 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 plume and

This material is based upon work supported by the National Science Foundation under Grant No. 1637915.

The authors are with the Department of Electrical and Computer Engineering, Virginia Tech, USA. {yooncs8,ddeeksha, tokekar}@vt.edu.

return back to the starting position. Our algorithm builds on the one presented by Higashikawa *et al*. [10] for exploring an unknown binary tree. We show how to reduce the problem of exploring the plume to that of exploring a binary tree. We first start with the simpler scenario where the plume is modeled as a 2D grid and then generalize it to the case where the plume boundary is any smooth (formally defined in Section III) 2D curve. 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 plume, the number of robots, and the relative speeds of the plume 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.

The rest of the paper[1] is organized as follows. We begin by introducing the related work in Section II. We describe the problem setup in Section III. Our proposed algorithm for a grid-based map is presented in Section IV. We then extend this to arbitrarily-shaped shape in Section V. We present results from representative simulations in Section VI and field experiments in Section VII, respectively, before concluding with a discussion of future work in Section VIII.

II. Related Work

Environmental monitoring has extensively been studied in robotics due to its practical applications. Some of highlighted tasks include precision agriculture [13], [14], wildlife habitat monitoring [15]–[17] and atmospheric plume tracking [18]–[20]. For survey results, see Reference [2]. 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 [21] listed coverage path planning algorithms that can be used for different sensing and motion models. In case of plume tracking, the aim is to explore and map a plume by robots with limited sensing capability.

The objective of online exploration [4]–[6] is to explore and map a region without having prior knowledge on the size and the shape of the region. Corah and Michael [22] studied informative exploration in which the objective is 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 [23]. Plonski *et al*. [17] 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*. [24] 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 [25] proposed a vision-based exploration and mapping solution for a single robot. Cesare *et al*. [26] 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*. [27] and Das *et al*. [28] 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 a plume 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 and Kamphans [29] proposed a strategy of generating a competitive tour for online milling which may contain holes. Icking [7] showed $\frac{4}{3}$–competitive algorithm for online milling without considering holes. The algorithms presented by Arkin *et al*. [30] have $(3+\epsilon)$–approximation for offline lawn mowing and 2.5–approximation for offline milling. Kolenderska *et al*. [8] 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*. [31] 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 FOV.

Previous works in computational geometry assume specific properties of the region under exploration to ease the analysis. We restrict the plumes to satisfy a specific notion of fatness (defined in the next section). Stappen and Overmars [32] used the notion of k–fatness in motion planning with obstacles — the smaller the value of k, the fatter the obstacle. Efrat [33] 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*. [34] adopted the same notation of the fatness for the application of triangulating and guarding polygons. Lee *et al*. [35] 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 plume shape.

[1]A preliminary version of this paper was presented at [12]. This version improves upon [12] with a more expansive literature survey, a more detailed explanation on the proposed algorithm, and new simulation and experimental results.

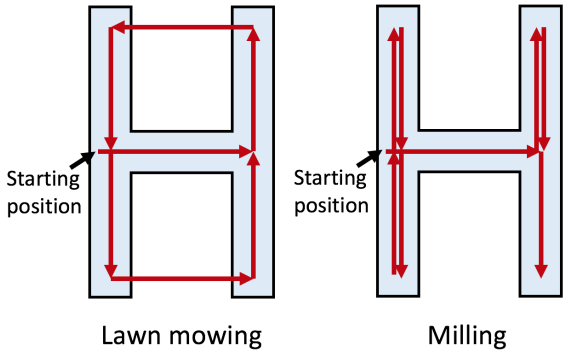


Fig. 2. Example of different trajectories obtained by applying lawn mowing and milling approaches to the H-shaped plume. Whereas lawn mowing allows the robot to move outside the plume, milling restricts the robot to stay inside the plume.

When multiple robots are considered, most of works [3], [9], [10], [36]–[38] have studied a tree-based exploration by employing a recursive Depth-First Search (DFS). In these works, the environment to be explored is assumed to be a tree. Fraigniaud *et al.* [9] proposed a tree exploration algorithm using R robots that is $\mathcal{O}(\frac{R}{\log R})$–competitive. In their work, each robot is allowed to observe the incident edges but not the adjacent vertices. Brass *et al.* [3] used the same sensing model and improved the competitive ratio of Fraigniaud *et al.* [9] 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.* [37] improved the lower bound proposed by Fraigniaud *et al.* [9] of $2-\frac{1}{R}$ to $\Omega(\frac{\log R}{\log\log R})$.

Megow *et al.* [39] 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.* [10] 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.* [40] presented an algorithm for minimizing the number of robots given limited energy E for each robot.

Preshant *et al.* [38] showed that the competitive ratio remains largely the same, $\frac{2(\sqrt{2}R+\log R)}{1+\log R}$, when the environment is an orthogonal polygon[2] but is 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.* [28] and Brass *et al.* [3] 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.* [35] proposed distributed online exploration algorithms assuming a fully connected network. In Higashikawa *et al.* [10], robots can communicate with each other when they meet at the same vertex. We adopt the same model.

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

III. Problem Description

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

We assume the plume is translating on the plane at zero height and that the aerial robots fly at a fixed altitude. Each robot has a downwards-facing camera that yields a square footprint on the plane containing the plume. That is the FOV is a square. Without loss of generality, we assume that the side length of the square FOV is 1 in this work.

We consider plumes whose size is at least as large as the FOV of the robots. Specifically, we require the plume to satisfy the following assumption.

Definition 1 (Fat Plumes). *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 plume P is* fat *if $B(p)$ lies completely inside $int(P)$ for all $p' \in \partial P$.*

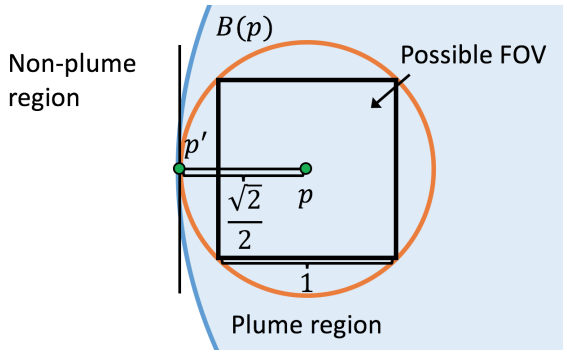


Fig. 3. We restrict our attention to plumes that are *fat* (Definition 1).

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

We assume that the plume translates with a fixed speed of S_p in a fixed direction, both of which are known to the robots.[3] The speed and the moving direction of the 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 [41].

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 plume. We seek tours for each robot that explore the plume and return back to this starting location.

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

Problem 1 (Multi-Robot Exploration of Translating Plume)**.** *Find a tour for all the robots that minimizes the exploration time such that every point in the plume is visible from the FOV 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.

IV. Plume Exploration over a Grid Map

In this section, we present our main algorithm. We first solve a simpler version of Problem 1 where the plume 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 plume 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 plume shape with a grid.

A. Recursive DFS Algorithm for a Grid Map

In this section, we assume that the plume is represented as a grid map [42]. 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 plume P is just a collection of C cells that form one connected set (if a cell $c \in P$ is part of the plume, then one of its four neighbors must also be a part of the plume when $C > 1$).

The problem of exploring the plume is then simplified to that of exploring a grid map and identify the cells that belong in P. Since we assume that the FOV 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 plume 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*. [10]. Higashikawa *et al*. [10] 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 plume. The neighboring cells that contain the plume 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 weight 0 and a dummy vertex, we can convert the tree into a binary tree without loss of generality.[4]

Each neighboring plume 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 plume cells but excluding the starting cell) at least once by one of the robots.

The weight of an edge is equal to the time taken by the robot to go from the center of one cell to that of the neighbor. Since we know the velocity with which the plume is translating, we can determine the location of the center of the neighboring cell at a future time instance using the relative velocities between the plume and 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 a plume 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.[5] 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*.

B. Theoretical Analysis

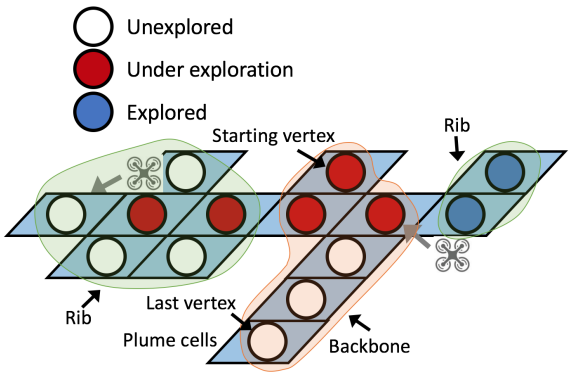
In this section we analyze the proposed Algorithm 1. We start with the following lemma which bounds the total weight of the tree generated by the robots.

Lemma 1 (Total Weight of the DFS Tree)**.** *Let L be the sum of the weights of all the edges in the tree generated by the recursive DFS algorithm. We have $L \leq \frac{C-1}{S_r - S_p}$.*

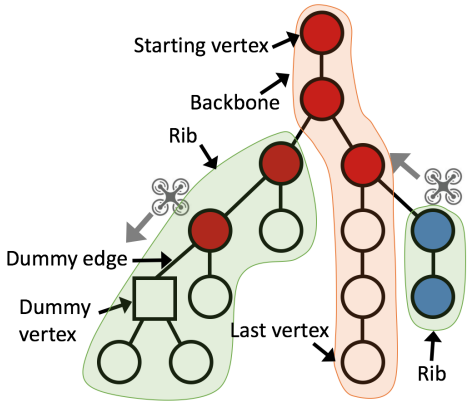
Proof. The DFS tree generated by Algorithm 1 contains C vertices that correspond to the plume cells and some number, say X, of dummy vertices. The incoming edges for all the dummy vertices have a weight of 0. Therefore, the total weight of the tree is only determined by the weight of the incoming edges of the $C-1$ vertices (the root vertex does not have an incoming edge). The weight of an edge is equal to the time taken by the robot to go between the centers of the respective cells. The cells are a unit distance apart from each other. However, the robot travels with a speed of S_r and the

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

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



(a) Two robots exploring the grid map.



(b) Tree generated from the recursive DFS.

Fig. 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*.

Algorithm 1: Multi-Robot Recursive DFS

```
Observe N(v) to determine whether neighboring cells are plume cells or non-plume cells.
if |N(v)|=0 then
    Mark v as explored.
    Move back to the parent vertex (→next vertex) and directly jump to Line 24.
end
Communicate with robots to update the state of N(v), i.e., unexplored, under exploration, and explored.
N(v) ← N(v)\{explored vertices}.
if v' ∈ N(v) is under exploration then
    if moving to v' generates a cycle in the tree then
        N(v) ← N(v)\{v'}.
    end
end
if |N(v)| > 1 then
    if |N(v)| > 2 then
        Add a dummy edge of weight 0 and a dummy vertex in order to keep the tree as a binary tree.
    end
    Split robots into two children as equally as possible.
    Move to one of two children (→next vertex) and mark v as under exploration.
else if |N(v)| = 1 then
    Move to the child (→next vertex) and mark v as under exploration.
else if |N(v)| = 0 then
    Move back to the parent vertex (→next vertex).
end
v ←the next vertex.
```

plume (thereby, the center of the cell) travels with a speed of S_p. The worst-case time taken by the robot to travel a unit distance occurs when the plume (*i.e.*, the center) is moving directly away from the robot. The minimum relative speed is $S_r - S_p$. Therefore, the maximum time taken for executing an edge is $\frac{1}{S_r - S_p}$. Since there are $C-1$ edges in the tree, the total weight will be less than $\frac{C-1}{S_r - S_p}$. □

a) Upper Bound Analysis: To analyze the cost of the proposed algorithm, we adapt the token collecting rule proposed by Higashikawa *et al.* [10] to the case of a translating plume. Note that this rule is not required for implementing the algorithm, but only for analyzing the competitive ratio.

Higashikawa *et al.* [10] define the concept of 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.

Higashikawa *et al.* [10] place two tokens of weight $w(e)$ on every rib edge e and $1 + \lfloor \log R \rfloor$ tokens of weight $w(e)$ on every backbone edge e. The weight of a token is set to be equal to the length of the corresponding edge denoted by $l(e)$, *i.e.*, $w(e) = l(e)$. Thus, the following relation can be obtained if the robot moves at unit speed:

$$w(e) = l(e) = \frac{\text{Unit length}}{\text{Unit time}} \times t(e) = \text{Unit speed} \times t(e) = t(e), \tag{1}$$

where $t(e)$ denotes the time taken for traversing the edge e. This implies that one token of unit weight (or unit length) per unit time (or, equivalently, one token of unit speed) can be collected. Let t_{last} be the time when the leaf vertex of the backbone is visited. They show that at least a total weight of $(1 + \lfloor \log R \rfloor) t_{last}$ tokens are collected in a tree. This is upper bounded by $2(L - d_{max}) + (1 + \lfloor \log R \rfloor) d_{max}$, where d_{max} denotes the distance of the farthest vertex in the tree from the root.

In our case, however, the robot speed is not unit speed but its relative speed with respect to the plume speed. The relative speed is lower bounded by $(S_r - S_p)$ and upper bounded by $(S_r + S_p)$. Therefore, the amount of tokens that can be collected must be multiplied by these relative speeds in order to handle a translating plume (*i.e.*, tree).

We call the group as a set of robots moving together in the tree. Then, the modified rule is as follows: (1) At least $(S_r - S_p)$ and at most $(S_r + S_p)$ tokens are collected by one robot in a group that visits an edge e in a rib in the forward direction for the first time. (2) At least $(S_r - S_p)$ and at most $(S_r + S_p)$ tokens are collected by one robot in a group that

visits an edge e in a rib in the backward direction for the first time. (3) At least $(S_r - S_p)$ and at most $(S_r + S_p)$ tokens are collected by each of $1 + \lfloor \log R \rfloor$ robots that move along a backbone edge e in the forward direction for the first time. (4) At least $(S_r - S_p)$ and at most $(S_r + S_p)$ tokens are collected by one robot in a group that move along a backbone edge e in the forward direction after the first group.

Let t_{last} be the time when the last robot reaches a leaf vertex in the tree. Then, based on the four observations above we have:

$$\begin{aligned}(S_r - S_p)(1 + \lfloor \log R \rfloor) t_{last} \leq \textit{Total amount of tokens} \\ \leq (S_r + S_p)(1 + \lfloor \log R \rfloor) t_{last}.\end{aligned} \tag{2}$$

The *total amount of tokens* is equal to $2(L - d_{max}) + (1 + \lfloor \log R \rfloor) d_{max}$.

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

Lemma 2 (Upper Bound for Multi-Robot Recursive DFS)**.**

$$ALG \leq \frac{2(C + d_{max} \lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \tag{3}$$

The proof is given in Appendix A.

Corollary 1 (Special Cases)**.** *Upper bounds for the following special cases can be derived from Lemma 2, such as Multi-Robot Static Plume (MRSP), Single Robot Translating Plume (SRTP), and Single Robot Static Plume (SRSP).*

MRSP	SRTP	SRSP
$\texttt{ALG} \leq \frac{2(C + d_{max} \lfloor \log R \rfloor)}{S_r(1 + \lfloor \log R \rfloor)}$	$\texttt{ALG} \leq \frac{2C}{S_r - S_p}$	$\texttt{ALG} \leq \frac{2C}{S_r}$

TABLE I
UPPER BOUNDS OF SPECIAL CASES.

Note that the upper bound for MRSP becomes the result from Higashikawa et al. [10] if $S_r = 1$. Also, the upper bound for SRSP is equivalent to Icking et al. [29] if $S_r = 1$.

Proof. Please refer to Appendix B. □

b) Lower Bound Analysis: We study the lower bound for the optimal algorithm in order to obtain a competitive ratio. Let $\texttt{OPT}_g^1$ 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:

$$\texttt{OPT}_g^1 \geq \frac{C - 1}{S_r + S_p}. \tag{4}$$

We use $\texttt{OPT}_g^R$ to represent the time taken by the optimal algorithm over any grid polygon of a plume region using R robots. Then, the following lemma gives the lower bound for $\texttt{OPT}_g^R$.

Lemma 3 (Lower Bound for Optimal Algorithm)**.**

$$OPT_g^R \geq \frac{C - 1}{(S_r + S_p)R}. \tag{5}$$

Proof. Please refer to Appendix C. □

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

$$\begin{aligned}ALG \leq & \frac{2(S_r + S_p)(R + \lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)} OPT_g^R \\ & + \frac{2}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}.\end{aligned} \tag{6}$$

Proof. Substituting Equation (5) into Equation (3) gives:

$$\texttt{ALG} \leq \frac{2((S_r + S_p)R\texttt{OPT}_g^R + 1 + d_{max} \lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \tag{7}$$

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

$$\leq \frac{2(S_r + S_p)(R + \lfloor \log R \rfloor)\texttt{OPT}_g^R + 2}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \tag{8}$$

□

V. PLUME EXPLORATION OVER AN ARBITRARY PLUME SHAPE

The presented results so far are for a grid map approximation of the plume. In this section, we will relate the bounds obtained for the grid map case to the case of arbitrarily-shaped plumes. Specifically, we will extend Lemma 2 to apply to a plume region that may have an arbitrary shape.

The algorithm for exploring the plume remains the same. We will still construct a tree that represents a grid map of the plume. The main difference here is that in the previous analysis, we assumed that the boundary of the plume 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 plume (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 plume whereas the inner grid map lies completely inside the plume. Therefore, the term C in the upper bound (Lemma 2) will now be replaced by C_{out}^{ALG}. However, the C term in the lower bound (Lemma 3) 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 plume.

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 4 (Grid Approximation of Arbitrary Plume Shape)**.** *The upper bound on C_{out}^{ALG} for a* fat *polygon (from Definition 1) is given by:*

$$C_{out}^{ALG} \leq 3C_{in}^{ALG} + 6. \tag{9}$$

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

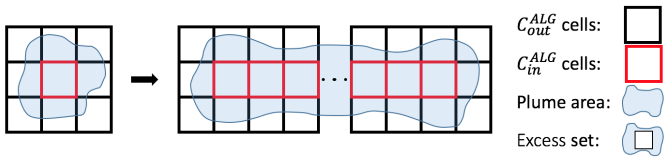


Fig. 5. Row formation of C_{in}^{ALG} cells as the number of cells changes from 1 to a finite number.

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[6] 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 (9). □

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 plume. 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 plume (and adding any other cell to C_{in}^{BEST} would not allow C_{in}^{BEST} to be completely inside the plume). The relationship is given by:

Lemma 5 (Best Possible Grid-Approximation)**.**

$$C_{in}^{ALG} \leq 6C_{in}^{BEST}. \tag{10}$$

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

[6]A vertex is called the reflex vertex if the external angle formed by two edges incident to this vertex is less than 180°.

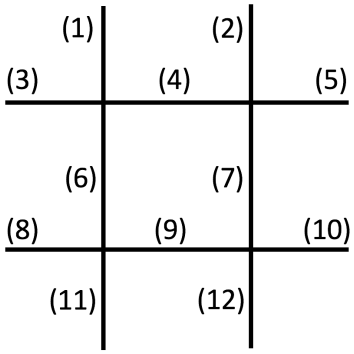


Fig. 6. A part of grid cell from any grid approximation. Unique number is assigned to a different side of grid cells.

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, $C_{in}^{ALG} \leq 6C_{in}^{BEST}$.

□

Finally we give our main result as follows:

Theorem 2 (Competitive Ratio for Arbitrary Plume Shape)**.** *Let* $\mathtt{OPT}^R$ *be the time taken by the optimal algorithm over any arbitrary plume shape using* R *robots.*

$$\begin{aligned} ALG \leq & \frac{2(S_r + S_p)(18R + \lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)} OPT^R \\ & + \frac{48}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \end{aligned} \tag{11}$$

Proof. Although $\mathtt{OPT}^R$ is the cost for any arbitrary plume shape, we can still lower bound this using C_{in}^{BEST} (similar to Lemma 3) as:

$$\mathtt{OPT}^R \geq \frac{C_{in}^{BEST} - 1}{(S_r + S_p)R}. \tag{12}$$

Let $(S_r - S_p)(1 + \lfloor \log R \rfloor)$ be $\mathcal{M}$. We can obtain the following inequalities from Lemmas 2, 4, and 5 as follows.

$$\mathtt{ALG} \leq \alpha C_{out}^{ALG} + a \leq \beta C_{in}^{ALG} + b \leq \gamma C_{in}^{BEST} + b, \tag{13}$$

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

Substituting Equation (12) into the last inequality of Equation (13) and using $\frac{d_{max}}{S_r + S_p} \leq \mathtt{OPT}^R$, we have:

$$\begin{aligned} \mathtt{ALG} &\leq \frac{36(S_r + S_p)R}{\mathcal{M}} \mathtt{OPT}^R + \frac{48 + 2d_{max}\lfloor \log R \rfloor}{\mathcal{M}}, \\ &\leq \frac{2(S_r + S_p)(18R + \lfloor \log R \rfloor)}{\mathcal{M}} \mathtt{OPT}^R + \frac{48}{\mathcal{M}}. \end{aligned} \tag{14}$$

□

VI. Simulation

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 plume (Theorem 1).

We randomly generated a set of plume grid maps. We randomly choose one of four directions (*i.e.*, north, south, east, and west) for the direction of translation of the plume. The assumption that the moving direction of the plume 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 plume 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 plume cells, the number of robots, and the speed ratio between the robot and the plume. The lower bound in our analysis is given in Lemma 3. In deriving the lower bound, we assume that the robots always travel opposite to the plume, 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 plume. The axis of the rectangle is aligned with the direction of translation of the plume. 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 plume. Nevertheless, we find that the exploration time for our algorithm is comparable to this lower bound.

Each case was obtained from 100 trials (see Figures 7 (b–d)). Figure 7 (b) shows that the expected exploration time for all cases is proportional to the number of plume cells. The difference between the maximum and minimum costs also becomes larger as more plume cells are to be explored. Figure 7 (c) plots the exploration time when changing the number of robots. The exploration time of our algorithm and the lower bound decrease as the number of robots increases. Unlike these, the upper bound does not show a steady decreasing tendency because randomly generated plume cells affect d_{max}. Figure 7 (d) shows the exploration time when changing the speed ratio between the robot and the plume, *i.e.*, $\frac{S_r}{S_p}$. The exploration time for our algorithm and the upper bound decrease 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.

VII. Field Experiment

In this section, we conduct proof-of-concept experiments using a single UAV equipped with a downward-facing 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 plume. 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 [43]. The onboard software controls the UAV, communicates with GoPro HERO4 camera over WiFi to read sensor information, and detects the runway.

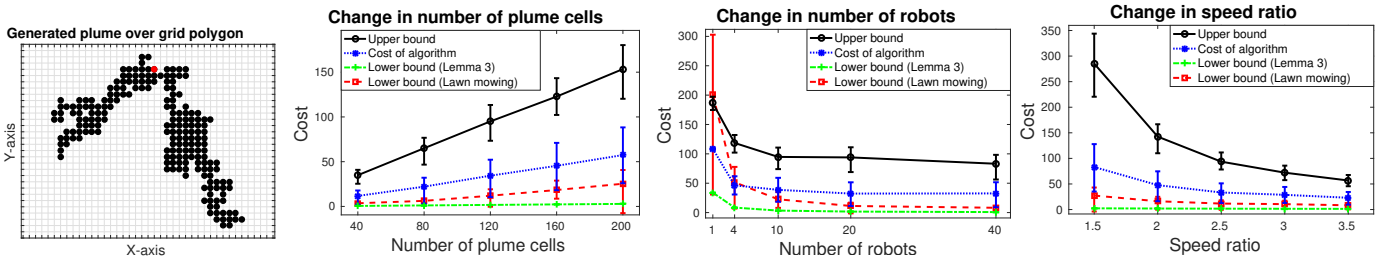
Our planning strategy consists of two modes: (1) lawn mowing and (2) the single-robot version of Algorithm 1. The input to the lawn-mowing mode is a bounding box that is guaranteed to contain at least some part of the runway. In the lawn-mowing mode, the UAV sweeps the bounding box. As soon as the UAV 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 [44]. 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 was not given to the UAV a priori.

Each cell was of size $4m \times 4m$. We used 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 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 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.

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



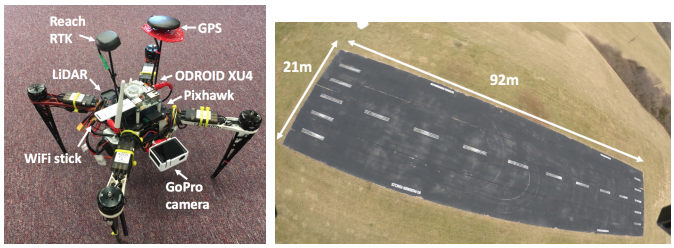
(a) Example of the randomly generated plume over grid cells. The red dot represents the starting vertex for robots.

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

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

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

Fig. 7. Simulation results. We fixed the number of plume 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.



(a) UAV platform.

(b) Runway ($92m \times 21m$) as a surrogate of the hazardous plume.

Fig. 8. Experimental setting.

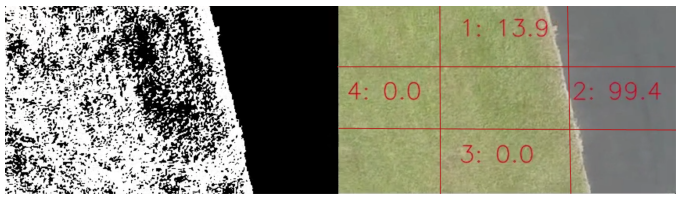


Fig. 9. Example of our sensing model using a single image that contains both the runway and the grassy region. The left image is the thresholded image. The right image shows the detection result indicating the percentage of black pixel values printed on the grid cells (colored in red in four neighboring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

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 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%.

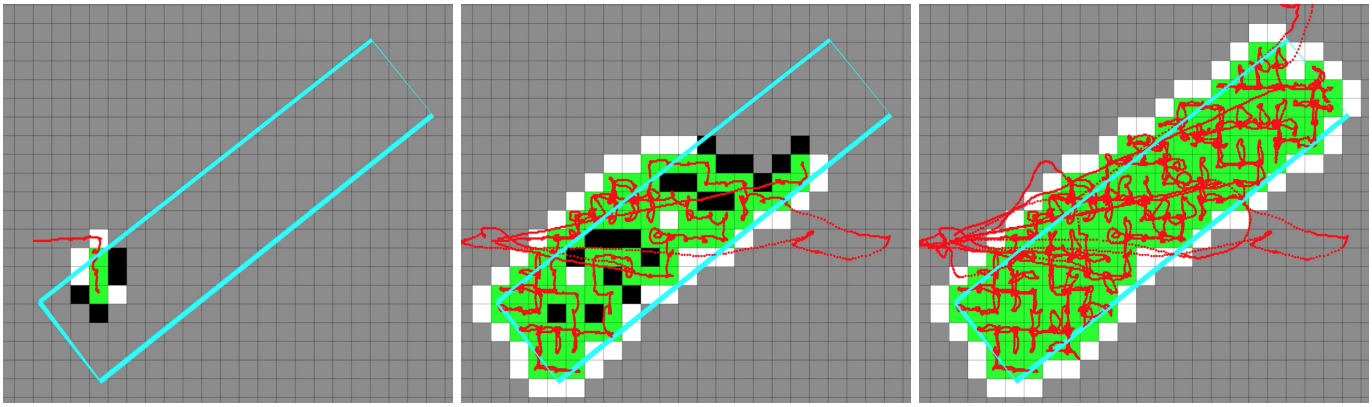
We also compute the false-positive and false-negative detection rates. Out of the total 483 detections, 27 were false positives and 53 were false negatives. Note that all but two cells that gave false-negative detections, eventually gave a positive detection. As a result, the final map (Figure 11) has 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.

VIII. Conclusion

We propose a recursive DFS algorithm for a team of aerial robots to explore a translating plume without knowing its shape and size. We present two approaches for the given problem where the first approximates the plume to map to the grid whereas the second considers any arbitrary shape of plume 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 plume.

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 plume 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



(a) Snapshot at time 2 minutes. (b) Snapshot at time 26 minutes and 40 seconds. (c) Snapshot at time 1 hour and 13 minutes.

Fig. 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, no plume, plume and explored cells, respectively. The total flight time taken to completely explore the entire runway was 1 hour and 13 minutes by using 6 batteries in a row. The video is available at: [Include the url].

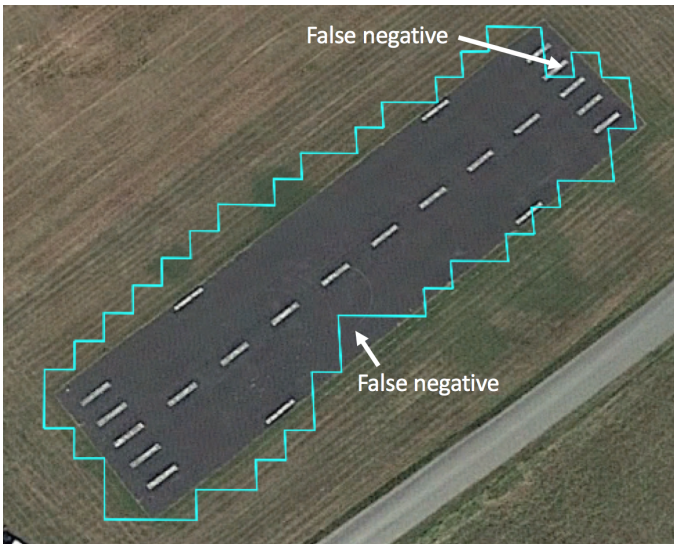


Fig. 11. Resultant boundary of the plume region (*i.e.*, the boundary of the grid map) plotted on Google Earth colored in blue.

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.

REFERENCES

[1] G. W. Podnar, J. M. Dolan, A. Elfes, S. Stancliff, E. Lin, J. C. Hosler, T. J. Ames, J. Moisan, T. A. Moisan, J. Higinbotham *et al.*, "Operation of robotic science boats using the telesupervised adaptive ocean sensor fleet system," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 1061–1068.

[2] M. Dunbabin and L. Marques, "Robots for environmental monitoring: Significant advancements and applications," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 24–39, 2012.

[3] P. Brass, F. Cabrera-Mora, A. Gasparri, and J. Xiao, "Multirobot tree and graph exploration," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 707–717, 2011.

[4] M. Juliá, A. Gil, and O. Reinoso, "A comparison of path planning strategies for autonomous exploration and mapping of unknown environments," *Autonomous Robots*, vol. 33, no. 4, pp. 427–444, 2012.

[5] S. Nuske, S. Choudhury, S. Jain, A. Chambers, L. Yoder, S. Scherer, L. Chamberlain, H. Cover, and S. Singh, "Autonomous exploration and motion planning for an unmanned aerial vehicle navigating rivers," *Journal of Field Robotics*, vol. 32, no. 8, pp. 1141–1162, 2015.

[6] Y. Girdhar, P. Giguere, and G. Dudek, "Autonomous adaptive exploration using realtime online spatiotemporal topic modeling," *The International Journal of Robotics Research*, vol. 33, no. 4, pp. 645–657, 2014.

[7] C. Icking, T. Kamphans, R. Klein, and E. Langetepe, "Exploring simple grid polygons," in *International Computing and Combinatorics Conference*. Springer, 2005, pp. 524–533.

[8] A. Kolenderska, A. Kosowski, M. Małafiejski, and P. Żyliński, "An improved strategy for exploring a grid polygon," in *International Colloquium on Structural Information and Communication Complexity*. Springer, 2009, pp. 222–236.

[9] P. Fraigniaud, L. Gasieniec, D. R. Kowalski, and A. Pelc, "Collective tree exploration," *Networks: An International Journal*, vol. 48, no. 3, pp. 166–177, 2006.

[10] Y. Higashikawa, N. Katoh, S. Langerman, and S.-i. Tanigawa, "Online graph exploration algorithms for cycles and trees by multiple searchers," *Journal of Combinatorial Optimization*, vol. 28, no. 2, pp. 480–495, 2014.

[11] A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*. cambridge university press, 2005.

[12] Y. Sung and P. Tokekar, "A competitive algorithm for online multi-robot exploration of a translating plume," *arXiv preprint arXiv:1811.02769*, 2018.

[13] P. Tokekar, J. Vander Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic uav and ugv system for precision agriculture," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1498–1511, 2016.

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

[15] P. Tokekar, D. Bhadauria, A. Studenski, and V. Isler, "A robotic system for monitoring carp in minnesota lakes," *Journal of Field Robotics*, vol. 27, no. 6, pp. 779–789, 2010.

[16] P. Tokekar, E. Branson, J. Vander Hook, and V. Isler, "Tracking aquatic invaders: Autonomous robots for monitoring invasive fish," *IEEE Robotics & Automation Magazine*, vol. 20, no. 3, pp. 33–41, 2013.

[17] P. A. Plonski, J. Vander Hook, C. Peng, N. Noori, and V. Isler, "Environment exploration in sensing automation for habitat monitoring," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 25–38, 2017.

[18] H. Ishida, Y. Wada, and H. Matsukura, "Chemical sensing in robotic applications: A review," *IEEE Sensors Journal*, vol. 12, no. 11, pp. 3163–3173, 2012.

[19] T. Lochmatter, E. A. Göl, I. Navarro, and A. Martinoli, "A plume tracking algorithm based on crosswind formations," in *Distributed Autonomous Robotic Systems*. Springer, 2013, pp. 91–102.

[20] M. Fahad, Y. Guo, B. Bingham, K. Krasnosky, L. Fitzpatrick, and F. A. Sanabria, "Robotic experiments to evaluate ocean plume characteristics and structure," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 6098–6104.

[21] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[22] M. Corah and N. Michael, "Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice," *Autonomous Robots*, vol. 43, no. 2, pp. 485–501, 2019.

[23] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligence Research*, vol. 34, pp. 707–755, 2009.

[24] G. Hitz, A. Gotovos, M.-É. Garneau, C. Pradalier, A. Krause, R. Y. Siegwart *et al.*, "Fully autonomous focused exploration for robotic environmental monitoring," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 2658–2664.

[25] R. Sim and J. J. Little, "Autonomous vision-based robotic exploration and mapping using hybrid maps and particle filters," *Image and Vision Computing*, vol. 27, no. 1-2, pp. 167–177, 2009.

[26] K. Cesare, R. Skeele, S.-H. Yoo, Y. Zhang, and G. Hollinger, "Multi-uav exploration with limited communication and battery," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2230–2235.

[27] M. A. Bender, A. Fernández, D. Ron, A. Sahai, and S. Vadhan, "The power of a pebble: Exploring and mapping directed graphs," *Information and computation*, vol. 176, no. 1, pp. 1–21, 2002.

[28] S. Das, P. Flocchini, S. Kutten, A. Nayak, and N. Santoro, "Map construction of unknown graphs by multiple agents," *Theoretical Computer Science*, vol. 385, no. 1-3, pp. 34–48, 2007.

[29] C. Icking, T. Kamphans, R. Klein, and E. Langetepe, "Exploring an unknown cellular environment." in *EuroCG*, 2000, pp. 140–143.

[30] E. M. Arkin, S. P. Fekete, and J. S. Mitchell, "Approximation algorithms for lawn mowing and milling," *Computational Geometry*, vol. 17, no. 1-2, pp. 25–50, 2000.

[31] S. Arya, S.-W. Cheng, and D. M. Mount, "Approximation algorithm for multiple-tool milling," *International Journal of Computational Geometry & Applications*, vol. 11, no. 03, pp. 339–372, 2001.

[32] A. F. van der Stappen and M. H. Overmars, "Motion planning amidst fat obstacles," in *Proceedings of the tenth annual symposium on Computational geometry*. ACM, 1994, pp. 31–40.

[33] A. Efrat, "The complexity of the union of (α,β)-covered objects," *SIAM Journal on Computing*, vol. 34, no. 4, pp. 775–787, 2005.

[34] G. Aloupis, P. Bose, V. Dujmović, C. Gray, S. Langerman, and B. Speckmann, "Triangulating and guarding realistic polygons," *Computational geometry*, vol. 47, no. 2, pp. 296–306, 2014.

[35] S. K. Lee, S. P. Fekete, and J. McLurkin, "Structured triangulation in multi-robot systems: Coverage, patrolling, voronoi partitions, and geodesic centers," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1234–1260, 2016.

[36] A. Mahadev, D. Krupke, S. P. Fekete, and A. T. Becker, "Mapping and coverage with a particle swarm controlled by uniform inputs," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1097–1104.

[37] M. Dynia, J. ŁopuszaŃski, and C. Schindelhauer, "Why robots need maps," in *International Colloquium on Structural Information and Communication Complexity*. Springer, 2007, pp. 41–50.

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

[39] N. Megow, K. Mehlhorn, and P. Schweitzer, "Online graph exploration: New results on old and new algorithms," *Theoretical Computer Science*, vol. 463, pp. 62–72, 2012.

[40] S. Das, D. Dereniowski, and C. Karousatou, "Collaborative exploration by energy-constrained mobile robots," in *International Colloquium on Structural Information and Communication Complexity*. Springer, 2015, pp. 357–369.

[41] J. Petrich and K. Subbarao, "On-board wind speed estimation for uavs," in *AIAA Guidance, Navigation, and Control Conference*, 2011, p. 6223.

[42] Z. A. Algfoor, M. S. Sunar, and H. Kolivand, "A comprehensive study on pathfinding techniques for robotics and video games," *International Journal of Computer Games Technology*, vol. 2015, p. 7, 2015.

[43] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

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

APPENDIX

A. Proof of Lemma 2

ALG can be upper bounded as follows:

$$\texttt{ALG} \leq t_{last} + \frac{d_{max}}{S_r - S_p}, \tag{15}$$

where t_{last} is the time that finishes visiting all vertices in the tree and $\frac{d_{max}}{S_r - S_p}$ is the time that traverses the longest length of the backbone when robot and plume moves away from each other. By substituting Equation (2) into the above equation, we have:

$$\leq \frac{2L + (\lfloor \log R \rfloor - 1) d_{max}}{(S_r - S_p)(1 + \lfloor \log R \rfloor)} + \frac{d_{max}}{S_r - S_p}. \tag{16}$$

Using Lemma 1, it becomes:

$$= \frac{2(C - 1 + d_{max} \lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \tag{17}$$

Removing a negative term completes the proof as:

$$\leq \frac{2(C + d_{max} \lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \tag{18}$$

B. Proof of Corollaries

The upper bound for MRSP can simply be obtained by plugging $S_p = 0$ into Equation (3) of Lemma 2.

The upper bound for SRTP can be derived from the upper bound of MRSP by having $R = 0$. However, we can even tighten the bound by using the following observation: if the robot and the plume 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:

$$\texttt{ALG} \leq \frac{C - 1}{S_r + S_p} + \frac{C - 1}{S_r - S_p}. \tag{19}$$

Taking out negative terms from the above equation becomes:

$$\leq \frac{2 S_r C}{(S_r + S_p)(S_r - S_p)}, \tag{20}$$

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 SRSP can be derived by plugging either $R = 1$ and $S_p = 0$ into the upper bound for MRSP or $S_p = 0$ into the upper bound for SRTP.

C. Proof of Lemma 3

We claim the following inequalities.

$$\texttt{OPT}^R \leq \texttt{OPT}^1, \tag{21}$$

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:

$$\texttt{OPT}^1 \leq R\texttt{OPT}^R, \tag{22}$$

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

Combining these inequalities and Equation (4), we prove Lemma 3.