A Geometric Approach for Multi-Robot Exploration in Orthogonal Polygons

Aravind Preshant Premkumar, Kevin Yu, and Pratap Tokekar*

Department of Electrical & Computer Engineering, Virginia Tech, USA.
{aravindp, klyu, tokekar}@vt.edu

Abstract. We present an algorithm to explore an orthogonal polygon using a team of p robots. Our algorithm is based on a single-robot polygon exploration algorithm and a tree exploration algorithm. We show that the exploration time of our algorithm is competitive (as a function of p) with respect to the offline optimal exploration algorithm. In addition to theoretical analysis, we discuss how this strategy can be adapted to real-world settings. We investigate the performance of our algorithm through simulations for multiple robots and experiments with a single robot. We conclude with a discussion of our ongoing work.

1 Introduction

Exploration of unknown environments using a single robot has been a well studied problem [1, 2]. The task can be performed much faster if multiple robots are used. The challenge is to come up with an algorithm to efficiently coordinate multiple robots to explore the environment in the minimum amount of time. Broadly speaking, there have been two types of approaches towards solving the exploration problem: geometric and information-theoretic. In geometric approaches (e.g., [3]), it is typical to assume that the robots have perfect, unrestricted and omni-directional sensing. Geometric algorithms typically give global guarantees on distance traveled at the expense of restrictive assumptions about the environment and sensor models. On the other hand, information-theoretic approaches (e.g., [4]) explicitly take into account practical constraints such as noisy sensors and complex environments. However, these approaches are often greedy (e.g., frontier-based [5]) and typically do not yield any guarantees on the total time taken. In this paper we investigate the challenges in applying geometric exploration algorithms to practical settings.

We use competitive analysis [6] in order to analyze the cost of exploration. Competitive ratio of an online algorithm is defined as the worst-case ratio (over all inputs) of the cost of the online algorithm and the optimal offline algorithm. The optimal offline algorithm corresponds to the case when the input (i.e., polygon) itself is known. The goal is to find online algorithms with low, constant

* This material is based upon work supported by the National Science Foundation under Grant No. 1566247.

To appear in the 12th International Workshop on Algorithmic Foundations of Robotics (WAFR), 2016

competitive ratios. That is, algorithms whose online performance is comparable to algorithms who know the input a priori. We focus on the case of exploring unknown orthogonal polygons[1] without any holes. Deng et al. [7] showed that there is an algorithm with a constant competitive ratio for exploring orthogonal polygons with a single robot. Our main result is a constant competitive ratio algorithm for exploring with p robots, when p is fixed (Section 3).

The analysis of this algorithm requires certain assumptions that do not necessarily hold in practice. Our second contribution is to show how to adapt this purely geometric algorithm for real-world constraints to incorporate sensing limitations and uncertainty (Section 4). We evaluate our algorithm through simulations and experiments on a mobile robot (Sections 5 and 6). We begin with a discussion of related work in computational geometry and robotics literature.

2 Related Work

In this section, we present the existing work related to the exploration problem. We organize the related work into three broad categories: polygon exploration, graph exploration, and information-theoretic exploration.

2.1 Exploration of Polygons

The study of geometric problems that are based on visibility is a well-established field within computational geometry. The classic problems are the art gallery problem [8], watchman route problem [9], and target search [10] and shortest path planning [11] in unknown environments.

Using a fixed set of positions for guarding a known polygonal region, i.e., a set of points from which the entire polygon is visible, is known as the classical art gallery problem. If the gallery is represented by a polygon (having n vertices) and the guards are points in the polygon, then visibility problems can be equivalently stated as problems of covering the gallery with star-shaped polygons. Chvatal [12] and Fisk [13] proved that $\lfloor n/3 \rfloor$ guards are always sufficient and sometimes necessary to cover a polygon of n edges. The minimum number of guards required for a specific polygon may be much smaller than this upper bound. However, Schuchardt and Hecker [14] showed that finding a minimum set of guards is NP-hard, even for the special case of an orthogonal polygon.

Finding the shortest tour along which one mobile guard can see the polygon completely is the watchman route problem. Chin and Ntafos[9] showed that the watchman route can be found in polynomial time in a simple rectilinear polygon. Exploring an unknown polygon is the online watchman route problem. Bhattacharya et al. [3] and Ghosh et al. [15] approached the exploration problem with discrete vision, i.e., they assume that the robot doesn't have continuous visibility and has to stop at different scan points in order to sense the

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

environment. They focus on the worst-case number of necessary scan points. Their algorithm results in a competitive ratio of $(r+1)/2$, where r is the number of reflex vertices in the polygon. For the case of a limited range of visibility they give an algorithm where the competitive ratio in a polygon P can be limited by $\lfloor \frac{8\pi}{3} + \frac{\pi R \times Perimeter(P)}{Area(P)} + \frac{(r+h+1)\pi R^2}{Area(P)} \rfloor$, where h is the number of holes in the polygon.

For a simple polygon, Hoffmann et al. [16] presented an algorithm which achieves a constant competitive ratio of $c = 26.5$. For the special case of an orthogonal polygon, Deng et al. [7] presented a $\sqrt{2}$ competitive exploration strategy for the case of a single robot. We show how to extend the single robot exploration algorithm by Deng et al. [7] to the case of p robots. The resulting algorithm has a competitive ratio that is a function of p.

2.2 Graph Exploration

The problem of visiting all the nodes in a graph in the least amount of time is known as the Traveling Salesperson Problem (TSP). Here, all nodes of the graph are known before-hand and the objective is to determine the shortest path visiting all the nodes in the graph exactly once. Finding the optimal TSP tour for a given graph is known to be NP-hard, even for the special case where the nodes in the graph represent points on the Euclidean plane [17]. For the Euclidean version of the problem, there exist polynomial time approximation schemes [17, 18], i.e., for any $\epsilon > 0$, there exists a polynomial time algorithm which guarantees an approximation factor of $1 + \epsilon$.

In the graph exploration version of the problem nodes are revealed in an online fashion. The objective is to minimize the total distance (or time) traveled. Fraigniaud et al. [19] presented an algorithm for exploration of trees using p robots with a competitive ratio of $\mathcal{O}(p/\log p)$ and a lower bound of $\Omega(2 + 1/p)$. This lower bound was improved by Dynia et al. [20] to $\Omega(\log p/\log\log p)$. They modeled the cost as the maximal number of edges traversed by a robot and presented a $(4 - 2/p)$-competitive online algorithm.

Higashikawa et al. [21] showed that greedy exploration strategies have an even stronger lower bound of $\Omega(p/\log p)$ and presented a $(p + \log p/1 + \log p)$ competitive algorithm.

Better bounds have been achieved for restricted graphs. Dynia et al. [22] presented an algorithm that achieves faster exploration for trees restricted by a density parameter k which forces a minimum depth for any subtree depending on its size. Trees embeddable in d-dimensional grids can be explored with a competitive ratio of $\mathcal{O}(d^{1-1/k})$. For 2-dimensional grids with only convex obstacles, Ortolf et al. improved the competitive ratio to $\mathcal{O}(\log^2 d)$ [23]. Despite these strong restrictions on the graph, the same lower bound of $\Omega(\log p/\log\log p)$ holds for all trees.

We show that the problem of exploring a polygon can be formulated as a multi-robot tree exploration problem.

2.3 Information-Theoretic Exploration

The geometric and graph-based approaches typically assume perfect sensing with no noise. In practice, however, we observe that measurements are not perfect and we have to account for noise in these measurements while employing exploration strategies. Exploration strategies can be broadly classified into two types: frontier based and information-gain based. We refer the reader to [2] for a comprehensive survey of exploration strategies.

Frontier based exploration strategies are primarily greedy in nature and drive the robots to a deterministic boundary between known and unknown spaces in the map. This strategy has been extended and used to perform exploration of unknown 2D [5, 24, 25] and 2.5D environments [26]. For a comprehensive survey please refer to Holz et al. [27].

Information-gain based strategies seek to optimize some information-theoretic measures such as minimizing the entropy of the map [28, 29], or maximizing mutual information [30, 31]. While these algorithms produce better maps, the planner is typically one-step greedy algorithms that do not have global guarantees on the total distance traveled. Charrow et al.[4] attempted to resolve this by combining a global planner for a single robot to determine goals whilst locally following a path which locally maximizes mutual information. Here, we present a global planner for multi-robot teams that has strong performance guarantees in the form of constant competitive ratio.

3 Multi-Robot Exploration Algorithm

In this section, we present the details of our algorithm for exploring an orthogonal polygon without any holes, P, using a team of p robots. Our algorithm builds on the algorithm by Deng et al. [7] for exploring an orthogonal polygon with a single robot and extends it to the case of multiple robots using the graph exploration strategy from [21]. Our main insight is to show that the path followed by the robot using the algorithm in [7] can be used to construct a tree, denoted by $\mathcal{T}$, in P. That is, exploring the polygon is equivalent to visiting all nodes in this tree. We show that a multi-robot tree exploration algorithm from [21] can be used to explore and visit every node in this tree. Furthermore, we show that the competitive ratio of our algorithm with respect to the optimal offline algorithm is bounded (as a function of p).

We assume all robots start at a common location. The cost of exploration is defined as the time taken for all the robots to return to the starting location having explored the polygon. We say that a polygon P is explored if all points in its interior and on the boundary were seen from at least one robot. For the purpose of the analysis, we assume that the sensor on the robot is an omni-directional camera with infinite sensing range which returns the exact coordinates of any object in its field of view. We also assume that the robots move at unit speeds and can communicate at all times. In the next section, we show how to adapt our algorithm to realistic sensing models and evaluate it through experiments.

We introduce some terminology used in our algorithm (refer to Figure 1) before presenting the details. The sub-polygon that is visible from a point x is known as the *visibility polygon* of x and is denoted by $VP(x)$. Some of the edges in the visibility polygon are part of the boundary of P where as others are chords in the interior of P (e.g., segment $\overline{gb}$ in Figure 1). A reflex vertex of P which breaks the continuity of the part of the boundary of P visible from x is known as a *blocking vertex*. Vertex b in Figure 1 is a blocking vertex. Let $\overline{bc}$ be the edge incident to b which is (partly) visible from x. The line segment perpendicular to $\overline{bc}$ drawn from b till the boundary of P is known as the *extension* of the blocking vertex b for an orthogonal polygon. The robot must cross the extension in order to "look beyond" the blocking vertex and explore the polygon. Draw the line segment starting from the robot's position x perpendicular to the extension $\overline{be}$. The point at which these two line segments intersect (E_g) is known as the *extension goal* corresponding to the blocking vertex b. An extension divides P into two sub-polygons. The one which contains the robot is known as the *home polygon* of the robot with respect to b. The other sub-polygon is known as the *foreign polygon* of the robot with respect to b and is denoted as $FP(b)$.

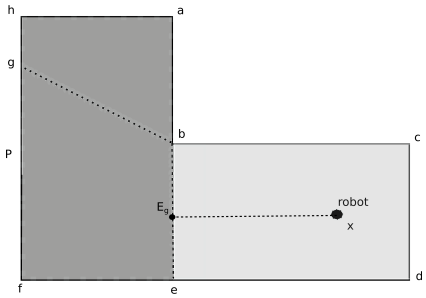

Fig. 1. Vertex b blocks the visibility of the robot and is known as a *blocking vertex*. The robot has to cross the segment $\overline{be}$ (known as the *extension*) to increase the visible boundary of the polygon P. The *extension* divides P into two sub-polygons. The sub-polygon not containing the robot (shaded in dark gray) is known as the *foreign polygon* w.r.t. b and is denoted as $FP(b)$.

The algorithm starts by creating a tree with the initial position of the robots as the root. All robots start in one cluster located at the root node. We use three labels to keep track of the status of any node in the tree: *unexplored*, *under-exploration*, and *explored*. Whenever a cluster of robots reach a node (using a subroutine `goto` not shown), we call the subroutine shown in Algorithm 1. While navigating using `goto`, if two clusters run into each other, then they merge and travel up the tree together.

The root is initially marked as *under-exploration*. We then check to determine any blocking vertices visible from the current node. We add the *extension goals* corresponding to any blocking vertices visible from the current node as its children. All of the corresponding extension goals are added as children by sorting them in the clockwise direction. These new children of the current node are adjusted according to the conditions mentioned. These conditions define an ordering over the nodes, by rewiring the tree. The cluster of robots at the current node is then divided as equally as possible and sent to visit the children of the current node. If the current node doesn't have any children, the current node

```
1  Function explore()
       Data: Cluster of robots, C, located at some node, A, in the tree.
2      if A is marked under-exploration then
3          A ← children of A not marked as explored;
4          if A == ∅ then
5              if A == root of the tree then
6                  Terminate exploration;
7              else
8                  goto(C,parent(A);
9              end
10         else
11             Divide C equally among A;
12             When any cluster reaches a child of A, call explore;
13         end
14     else
15         Mark A as under-exploration;
16         if blocking vertices detected from A then
17             Sort in clockwise direction and add as children of A;
18             for p and q are distinct blocking vertices do
19                 if FP(p) ⊆ FP(q) then
20                     add p as child of q;
21                 else
22                     if FP(p) and FP(q) intersect then
23                         add the vertex that appears first in the clockwise order
                           as the parent of the vertex that appears later;
24                     end
25                 end
26             end
27             Divide C equally among children of A;
28             When any cluster reaches a child of A, call explore;
29         else
30             Mark A as explored;
31             goto(C,parent(A);
32         end
33     end
```

Algorithm 1: Multi-Robot Exploration Subroutine

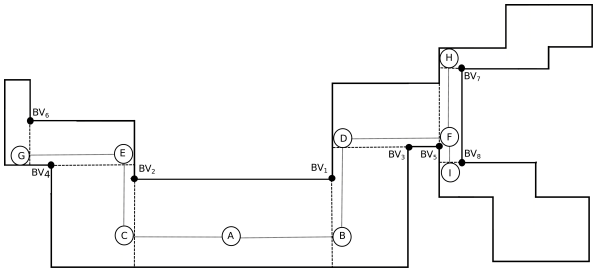

Fig. 2. An intermediate stage of exploration of a polygon by a team of 4 robots. Two robots located at the location G and the other two are located at F.

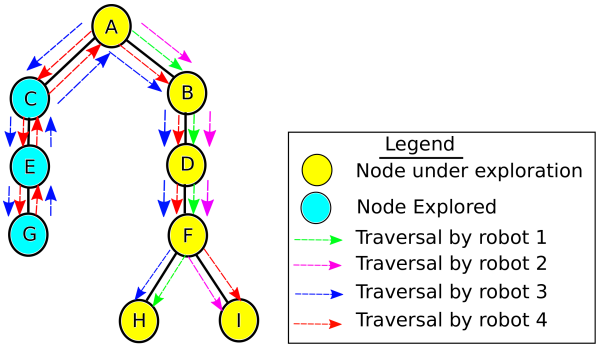

Fig. 3. The tree corresponding to the stage of exploration shown in Figure 2.

is marked as *explored*. The cluster moves to the parent of the current node to explore any of its other children which are *unexplored* or *under-exploration*. If a node does not have any children that are *unexplored* or *under-exploration*, then that node and its sub-tree is said to be explored. The exploration is said to be completed if the root of the tree is marked as *explored*.

Consider an example where P has been explored partially by a team of four robots as shown in Figure 2. All four robots $\{r_1, r_2, r_3, r_4\}$ start off at the location A which is added as the root of the tree as shown in Figure 3. Node A is marked as *under-exploration*. The robots observe two blocking vertices, BV_1 and BV_2. The extension goals B and C corresponding to BV_1 and BV_2, respectively, are added as the children of A in the tree. The robots split into two clusters $\{r_1, r_2\}$ and $\{r_3, r_4\}$. Cluster $\{r_1, r_2\}$ moves towards B and cluster $\{r_3, r_4\}$ moves towards C. The corresponding nodes in the tree are marked as *under-exploration*. At B, cluster $\{r_1, r_2\}$ observes a new blocking vertex, BV_3, and the corresponding extension goal D is added as the child of B. Since there is only one child, the cluster does not split and both robots move towards D. The corresponding node, D, in the tree is marked as *under-exploration*.

At C, the cluster $\{r_3, r_4\}$ observes a blocking vertex, BV_3, and the corresponding extension goal E is added as the child of C. Similarly, F is added as

the child of D and G is added as the child of E. At F, cluster $\{r_1, r_2\}$ observes two blocking vertices, the cluster splits into two clusters $\{r_1\}$ and $\{r_2\}$. Cluster $\{r_1\}$ moves towards H and cluster $\{r_2\}$ moves towards I. At G, cluster $\{r_3, r_4\}$ observes that there are no more blocking vertices. Hence, no children are added to G. G is marked as *explored* and the algorithm checks the predecessor in the tree, E. Since E does not have any other children for exploration, E is marked as *explored* as well.

Similarly, the algorithm checks its predecessor, C, and marks it as *explored*. Now the algorithm checks A, it has a child B which is *under-exploration*. Since there is no blocking vertex, the algorithm proceeds to check B. Similarly, the algorithm proceeds to check D and subsequently F. At F, there are two blocking vertices since neither of the two clusters , $\{r_1\}$ or $\{r_2\}$, have reached their goals. Thus, cluster splits into two $\{r_3\}$ and $\{r_4\}$. H is assigned to $\{r_3\}$ and I is assigned to $\{r_4\}$. The exploration algorithm proceeds in this manner up until the root is marked as *explored*.

3.1 Competitive Ratio Analysis

In this section, we prove that the competitive ratio of our algorithm is bounded with respect to the offline optimal algorithm. We divide our analysis into three steps. First, we show that the paths followed by all the robots can be mapped to navigating on a tree. Next we bound the sum of the costs of edges in the the tree with respect to the offline optimal cost. Finally, we bound the cost of our algorithm with respect to the cost of the tree.

Lemma 1. *The graph created in the exploration algorithm is a tree.*

Proof. We add an edge between two nodes when a new *extension goal* is observed due to a blocking vertex visible from the current location of the robot. Suppose for contradiction that graph created by the robots during exploration is not a tree. Hence, there exists a cycle in the graph. Draw the closed polygon corresponding to this cyclic path inside P. This polygon contains at least one blocking vertex (in fact, all blocking vertices corresponding to the cycle) in its interior. Furthermore, the boundary of this polygon does not intersect with the boundary of P. Therefore, there is an obstacle (hole) in the interior of P. This contradicts with our assumption that P is a simple, simply-connected orthogonal polygon. Hence, this graph is in fact a tree.

When $p = 1$, the proposed algorithm is the same as the one given by Deng et al. [7] for orthogonal polygons without holes. They showed that the competitive ratio of this algorithm is $\sqrt{2}$. Let C_{OPT} denote the time taken by the optimal algorithm for a single robot to explore P. Let C_{RECT} denote the time taken by the strategy from [7] for a single robot to explore P. We have $C_{\mathrm{RECT}} \leq \sqrt{2}C_{\mathrm{OPT}}$ from Theorem 3 in [7] and the assumption that the robots travel with unit speeds. Let C^p_{OPT} be the time taken by the optimal p robot exploration algorithm, and C_{ALG} be the time taken by the proposed algorithm. Our goal is to show an upper bound for $C_{\mathrm{ALG}}/C_{\mathrm{OPT}}$. We will show this by relating both quantities with C_{TREE} which is the sum of the lengths of edges in the tree.

Lemma 2. *$C_{TREE} \leq C_{RECT} \leq \sqrt{2} C_{OPT}$.*

Proof. The inequality $C_{\mathrm{RECT}} \leq \sqrt{2} C_{\mathrm{OPT}}$ holds from Theorem 3 of [7] for simple rectilinear (orthogonal) polygons. The inequality $C_{\mathrm{TREE}} \leq C_{\mathrm{RECT}}$ holds because any robot will have to back track on its path to reach previously unexplored areas and return back to the root.

Lemma 3. *If p robots are used to explore P and C^p_{OPT} is the cost of the optimal offline algorithm, then $C_{TREE} \leq \sqrt{2} C_{OPT} \leq \sqrt{2} p C^p_{OPT}$.*

Proof. Given the optimal algorithm for p robots, one can construct a tour for a single robot that executes each of the p tours. The length of such a tour is upper bounded by pC^p_{OPT}. Since C_{OPT} is the optimal cost for a single robot's tour, we have $C_{\mathrm{OPT}} \leq C^p_{\mathrm{OPT}}$. The other inequality follows from the previous lemma.

Theorem 1. *If C_{ALG} is the cost of exploring the polygon using the proposed strategy and C_{OPT} is the cost of exploring the polygon using an optimal offline strategy, then we have,*

$$\frac{C_{ALG}}{C_{OPT}} \leq \frac{2(\sqrt{2}p + \log p)}{1 + \log p} \tag{1}$$

Proof. The cost of exploring a tree with a recursive depth-first strategy used in the proposed algorithms is given by:

$$C_{\mathrm{ALG}} \leq \frac{2(C_{\mathrm{TREE}} + d_{\max} \log p)}{1 + \log p}, \tag{2}$$

where $d_{\max}$ is the maximum distance of a leaf node from the root in the tree. This bound comes directly from the result in [21]. In our case, $d_{\max}$ corresponds to the maximum distance of any extension goal from the starting position of the robots. It is easy to see that $d_{\max} \leq C^p_{\mathrm{OPT}}$.

From Lemma 3, we have:

$$C_{\mathrm{ALG}} \leq \frac{2(\sqrt{2} p C^p_{\mathrm{OPT}} + C^p_{\mathrm{OPT}} \log p)}{1 + \log p}$$

which yields

$$\frac{C_{\mathrm{ALG}}}{C_{\mathrm{OPT}}} \leq \frac{2(\sqrt{2}p + \log p)}{1 + \log p} \tag{3}$$

Thus, we show that the competitive ratio is bounded as a function of p. Figure 4 shows a plot of this bound as a function of p. We note that while the analysis only holds for the case of an orthogonal polygon without holes, the resulting algorithm can also be applied for polygons with holes. However, in such a case, the underlying graph created by the robots is not guaranteed to be a tree. Consequently, we would have to apply a bound for exploring general graphs with multiple robots to yield a similar competitive ratio. In the simulations, we show the empirical performance of our algorithm in environments with holes.

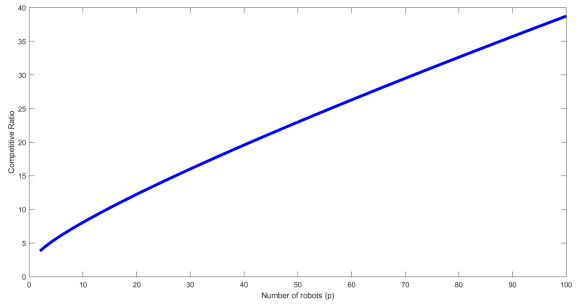

Fig. 4. Plot of the growth of competitive ratio as a function of the number of robots.

4 Adapting to Real-World Scenarios

Some of the assumptions made for the analysis do not hold in most practical scenarios. In this section, we show how to extend our basic algorithm framework in order to incorporate real-world constraints.

The main assumption is that of unlimited sensing range. In practice, the robot has a limited sensing range. For example, the robot cannot sense a long corridor with a single observation. Thus, in addition to blocking vertices, the robot has to move to *frontiers* at the end of its sensing range to sense more of the environment. This increases the distance the robot would have to cover compared to the distance it would have to cover if it had an infinite sensor. The robot thus has to detect two types of frontiers, one due to blocking vertices and the other due to sensing range. The only change to the algorithm is in Line 1 where we check for blocking vertices as well as frontiers due to sensing range.

We first detect blocking vertices in a given scan (as described below). Then frontier cells are clustered together to form frontiers due to sensing range. Any frontier which has a constituent frontier cell neighboring a blocking vertex is then discarded. For a blocking vertex, the *extension goal* is added on its *extension* with a slight offset. For frontiers due to sensing range, the middle frontier cell, after clustering, is chosen as the *frontier goal*.

Due to the sensing uncertainty, we represent the map built by the robots as a 2D occupancy grid (OG) as opposed to a geometric map. An OG is a discretized representation of the environment where each cell represents the probability of that space being occupied. Figure 5-right shows a representative OG. Cells with a lower probability of occupancy (< 0.5) are designated as free cells (represented as white in the OG) and cells with a higher probability of occupancy (> 0.5) are designated as occupied cells (represented as black in the OG). Cells which have not been observed are marked as unknown (represented as gray in the OG).

Typical sensors such as cameras and laser range finders have finite angular resolution. Consider three rays from the laser as shown in Figure 5-left. The rays intersect obstacles at the cells marked as black and all the cells the ray intersect between the robot (marked in blue) and the cell are marked as free. Due to the

finite resolution of the laser (0.395° for the hokuyo laser used in our system), the gray cells, even though they are in the field of the laser, are unobserved and this leads to gaps in observations. This leads to false frontiers being detected and hence such erroneous frontiers have to be discarded. We employ a simple heuristic of checking the size of a candidate frontier and discard those below a threshold.

Consider the robot (and the laser) located at the blue circle in Figure 5-right. The green ray represents one of the laser rays. In order to check for blocking vertices, occupied cells with a neighboring frontier cell are shortlisted first. In the figure, the cells marked with yellow and red *'X'* are identified. A blocking vertex, as defined earlier, is a reflex vertex. We can detect a reflex vertex in an OG by checking its four neighbors. If the four neighbors are an occupied cell, an unknown cell, a free cell which a frontier, and a free cell which is not a frontier we mark it as a blocking vertex. In Figure 5-right, the cell marked with the yellow *'X'* is detected as a blocking vertex.

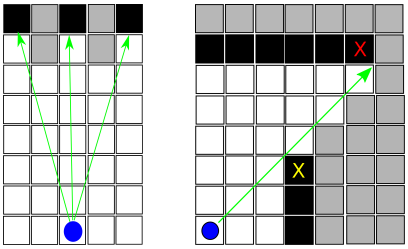

Fig. 5. (Left) Finite resolution of the laser leads to gaps in observations. **(Right)** A blocking vertex (marked with yellow 'X') has four distinctive neighbors: an occupied cell, an unknown cell, a free cell which is a frontier, one which is not.

Our algorithm also works for environments which are not orthogonal when occupancy grids are used as the underlying representation. Occupancy grids, typically, are orthogonal polygons by construction. Furthermore, for environments with holes (i.e., obstacles) the algorithm would create a tree and explore this tree. While this is correct, the distance traveled by the robots can be much higher than the optimal cost and as such the competitive ratio does not hold.

In the next section, we evaluate the empirical performance of our algorithm through simulations in such scenarios.

5 Simulations

We implemented our algorithm using ROS [32] and carried out simulations in Gazebo [33] in order to verify the correctness of the exploration algorithm in realistic environments. The five gazebo simulation environments used (Figure 6) are not all orthogonal and simply-connected – assumptions required for the analysis. The simulated robot is a differential-drive Pioneer P3-DX robot with a 2D Hokuyo laser range finder with a maximum range of 5 meters. The robot is localized in the environment using the *amcl* [34] package using a predefined map. Mapping during exploration is done using the octomap ROS package. The laser scan is converted into a point cloud which is then fed as input

to the *octomap* [35] node. Our implementation is available online at `https://github.com/raaslab/Exploration`.

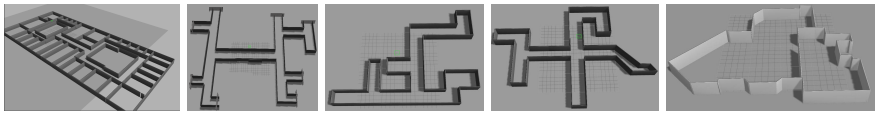

Fig. 6. Simulation environments 1 – 5 from left to right.

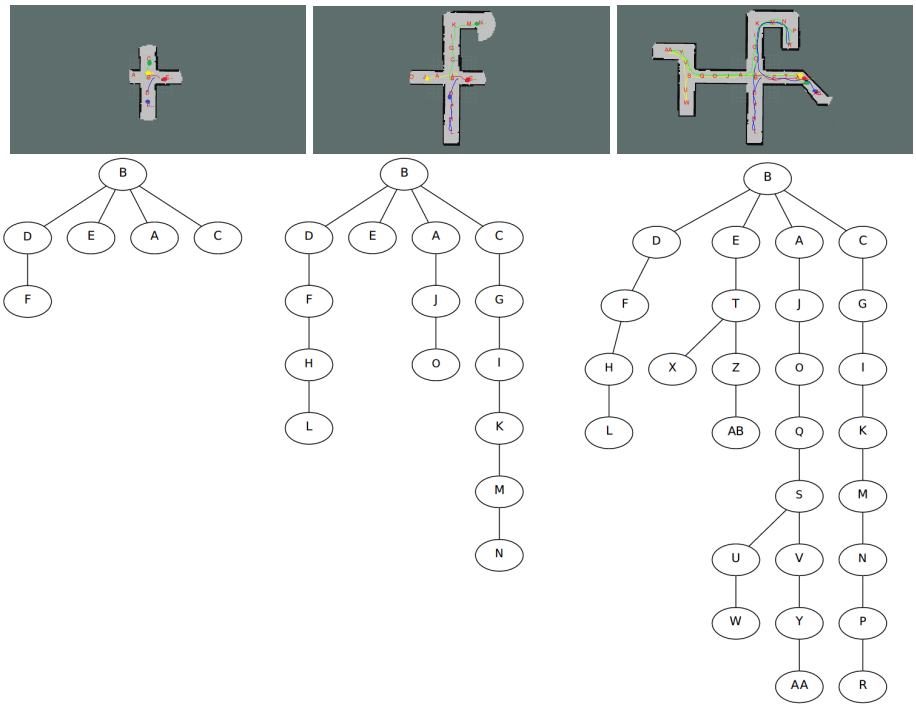

Fig. 7. Various stages while exploring environment 3 using four robots. The map explored and built along with the corresponding trees are shown.

Figure 7 shows various stages of exploration with four robots. The figure also shows the partial exploration tree built by the algorithm. The final trees produced while exploring all the environments is given in Figure 8. Table 1 shows the maximum distance traveled by a robot (in meters) during exploration for all the environments.

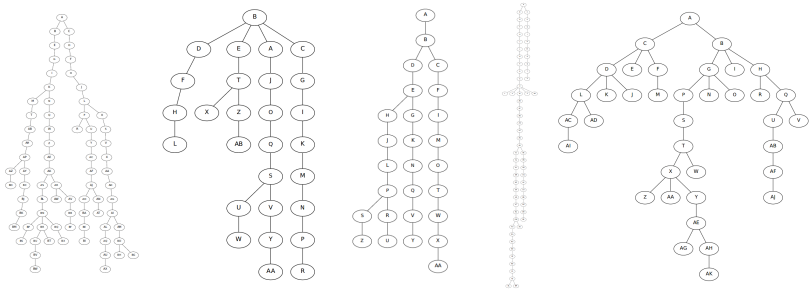

Fig. 8. Final tree produced after exploring the five environments in Figure 6 with four robots. The maximum distance traveled by a robot during exploration is given in Table 1.

	Env. 1	**Env. 2**	**Env. 3**	**Env. 4**	**Env. 5**
1 Robot	214.11	362.29	124.77	135.62	156.03
2 Robots	121.95	223.50	76.87	82.69	74.50
4 Robots	127.78	152.18	83.39	63.51	64.36

Table 1. Maximum distance traveled by a robot to explore environments in Fig. 6.

The cost of exploring environments 1, 3, 4, and 5 remains almost the same when the number of robots is increased from two to four. This is due to the fact that the exploration tree is not a balanced tree (Figure 8). On the other hand, in environment 2, the tree contains four or more *under-exploration* branches at all times. Consequently, the cost of exploration decreases significantly when four robots are used as opposed to just two.

6 Experiments

We carried out experiments using a Pioneer P3-DX robot mounted with a Hokuyo URG-04LX-UG01 2D laser and a Kinect2 RGBD sensor (Figure 9-left). The laser was configured to use 180° field of view and a resolution of 0.395°. During the exploration experiments, the robot used the 2D laser for localization using on a pre-built map. The map was generated using RTAB-map [36, 37] and the localization was carried out using the amcl package from ROS. Note that having a pre-built map is not a requirement for our algorithm. The amcl localization component could be replaced by, for example, any SLAM implementation. Furthermore, the robots generated a new map during the exploration process using octomapping [38] with the Kinect2 sensor. This map (and not the pre-built map) was used as the basis for finding blocking vertices in the proposed exploration algorithm.

The mapping and localization was performed on the pioneer P-3DX robot with two onboard computers in a master/slave configuration. The slave i7 Intel

Fig. 9. (Left) Pioneer P3-DX with the Hokuyo laser used for the experiments. The 2D Hokuyo laser was used for robot localization and the Kinect2 sensor was used for mapping during exploration. **(Right)** Final map as well as the exploration path during the corridor experiment.

NUC was dedicated to processing the Kinect2 data and the master i7Intel NUC was dedicated to running the localization and exploration algorithm. Figure 9-right shows the results of an exploration experiment in a corridor environment along with the path followed by the robot. A video of the system in operation is available online at `https://github.com/raaslab/Exploration`

7 Conclusion

We presented an algorithm for exploring an unknown polygonal environment using a team of p robots in the least amount of time. Our main theoretical contribution was to show that if the underlying environment is an orthogonal polygon without holes then our algorithm yields a constant competitive ratio for fixed p. Next, we showed how to adapt our algorithm so that it can extend to real-world sensing constraints. We verified the behavior of our algorithm through simulations in five representative environments with up to four robots and experiments with a single robot. We are currently working on experiments with multiple robots.

Future work includes extending our analysis to more general environments. Handling general polygons without holes, not necessarily orthogonal, is a direct extension of the algorithm presented here. The notion of blocking vertices remains the same and the underlying graph will still be a tree. However, the *extension goal* corresponding to the blocking vertex needs to be carefully defined. An immediate avenue of future work is to leverage the algorithm from [7] that allows for obstacles in orthogonal environments. For polygons with holes, the underlying graph is no longer a tree. Hence, a general graph exploration algorithm would have to be used.

References

1. Rao, N.S., Kareti, S., Shi, W., Iyengar, S.S.: Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms. Technical report, Citeseer (1993)
2. Juliá, M., Gil, A., Reinoso, O.: A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. Autonomous Robots **33**(4) (2012) 427–444
3. Bhattacharya, A., Ghosh, S.K., Sarkar, S.: Exploring an unknown polygonal environment with bounded visibility. In: International Conference on Computational Science, Springer (2001) 640–648
4. Charrow, B., Kahn, G., Patil, S., Liu, S., Goldberg, K., Abbeel, P., Michael, N., Kumar, V.: Information-theoretic planning with trajectory optimization for dense 3d mapping. In: Proceedings of Robotics: Science and Systems. (2015)
5. Yamauchi, B.: A frontier-based approach for autonomous exploration. In: Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on, IEEE (1997) 146–151
6. Motwani, R., Raghavan, P.: Randomized algorithms. Chapman & Hall/CRC (2010)
7. Deng, X., Kameda, T., Papadimitriou, C.: How to learn an unknown environment. i: the rectilinear case. Journal of the ACM (JACM) **45**(2) (1998) 215–245
8. O'Rourke, J.: Art gallery theorems and algorithms. Oxford University Press Oxford (1987)
9. Chin, W.p., Ntafos, S.: Optimum watchman routes. Information Processing Letters **28**(1) (1988) 39–44
10. Fleischer, R., Kamphans, T., Klein, R., Langetepe, E., Trippen, G.: Competitive online approximation of the optimal search ratio. SIAM Journal on Computing **38**(3) (2008) 881–898
11. Papadimitriou, C.H., Yannakakis, M.: Shortest paths without a map. Theoretical Computer Science **84**(1) (1991) 127–150
12. Chvatal, V.: A combinatorial theorem in plane geometry. Journal of Combinatorial Theory, Series B **18**(1) (1975) 39–41
13. Fisk, S.: A short proof of chvátal's watchman theorem. Journal of Combinatorial Theory, Series B **24**(3) (1978) 374
14. Schuchardt, D., Hecker, H.D.: Two np-hard art-gallery problems for ortho-polygons. Mathematical Logic Quarterly **41**(2) (1995) 261–267
15. Ghosh, S.K., Burdick, J.W., Bhattacharya, A., Sarkar, S.: Online algorithms with discrete visibility-exploring unknown polygonal environments. IEEE robotics & automation magazine **15**(2) (2008) 67–76
16. Hoffmann, F., Icking, C., Klein, R., Kriegel, K.: The polygon exploration problem. SIAM Journal on Computing **31**(2) (2001) 577–600
17. Mitchell, J.S.: Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric tsp, k-mst, and related problems. SIAM Journal on Computing **28**(4) (1999) 1298–1309
18. Arora, S.: Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. Journal of the ACM (JACM) **45**(5) (1998) 753–782
19. Fraigniaud, P., Gasieniec, L., Kowalski, D.R., Pelc, A.: Collective tree exploration. Networks **48**(3) (2006) 166–177
20. Dynia, M., ŁopuszaŃski, J., Schindelhauer, C.: Why robots need maps. In: International Colloquium on Structural Information and Communication Complexity, Springer (2007) 41–50

21. Higashikawa, Y., Katoh, N., Langerman, S., Tanigawa, S.i.: Online graph exploration algorithms for cycles and trees by multiple searchers. Journal of Combinatorial Optimization **28**(2) (2014) 480–495
22. Dynia, M., Kutyłowski, J., auf der Heide, F.M., Schindelhauer, C.: Smart robot teams exploring sparse trees. In: International Symposium on Mathematical Foundations of Computer Science, Springer (2006) 327–338
23. Ortolf, C., Schindelhauer, C.: Online multi-robot exploration of grid graphs with rectangular obstacles. In: Proceedings of the twenty-fourth annual ACM symposium on Parallelism in algorithms and architectures, ACM (2012) 27–36
24. Burgard, W., Moors, M., Stachniss, C., Schneider, F.E.: Coordinated multi-robot exploration. IEEE Transactions on robotics **21**(3) (2005) 376–386
25. Schwager, M., Dames, P., Rus, D., Kumar, V.: A multi-robot control policy for information gathering in the presence of unknown hazards. In: Proceedings of International Symposium on Robotics Research, Aug. (2011)
26. Cesare, K., Skeele, R., Yoo, S.H., Zhang, Y., Hollinger, G.: Multi-uav exploration with limited communication and battery. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE (2015) 2230–2235
27. Holz, D., Basilico, N., Amigoni, F., Behnke, S.: A comparative evaluation of exploration strategies and heuristics to improve them. In: ECMR. (2011) 25–30
28. Whaite, P., Ferrie, F.P.: Autonomous exploration: Driven by uncertainty. IEEE Transactions on Pattern Analysis and Machine Intelligence **19**(3) (1997) 193–205
29. Moorehead, S.J., Simmons, R., Whittaker, W.L.: Autonomous exploration using multiple sources of information. In: Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on. Volume 3., IEEE (2001) 3098–3103
30. Julian, B.J., Karaman, S., Rus, D.: On mutual information-based control of range sensing robots for mapping applications. The International Journal of Robotics Research (2014) 0278364914526288
31. Amigoni, F., Caglioti, V.: An information-based exploration strategy for environment mapping with mobile robots. Robotics and Autonomous Systems **58**(5) (2010) 684–699
32. Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, A.: ROS: an open-source Robot Operating System. In: ICRA Workshop on Open Source Software. (2009)
33. Koenig, N., Howard, A.: Design and use paradigms for gazebo, an open-source multi-robot simulator. In: Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on. Volume 3., IEEE (2004) 2149–2154
34. : AMCL ROS Packagel. `http://wiki.ros.org/amcl` Accessed: 2016-10-30.
35. Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., Burgard, W.: Octomap: An efficient probabilistic 3d mapping framework based on octrees. Autonomous Robots **34**(3) (2013) 189–206
36. Labbé, M., Michaud, F.: Online global loop closure detection for large-scale multi-session graph-based slam. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE (2014) 2661–2666
37. Labbe, M., Michaud, F.: Appearance-based loop closure detection for online large-scale and long-term operation. IEEE Transactions on Robotics **29**(3) (2013) 734–745
38. Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., Burgard, W.: Octomap: An efficient probabilistic 3d mapping framework based on octrees. Autonomous Robots **34**(3) (2013) 189–206