Competitive Algorithms and System for Multi-Robot Exploration of Unknown Environments

Aravind Preshant Premkumar

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Aug 1, 2017
Blacksburg, Virginia

Competitive Algorithms and System for Multi-Robot Exploration of Unknown Environments

Aravind Preshant Premkumar

(ABSTRACT)

We present an algorithm to explore an orthogonal polygon using a team of p robots. This algorithm combines ideas from information-theoretic exploration algorithms and computational geometry based exploration algorithms. The 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. We discuss how this strategy can be adapted to real-world settings to deal with noisy sensors. In addition to theoretical analysis, we investigate the performance of our algorithm through simulations for multiple robots and experiments with a single robot.

Competitive Algorithms and System for Multi-Robot Exploration of Unknown Environments

Aravind Preshant Premkumar

(GENERAL AUDIENCE ABSTRACT)

In applications such as disaster recovery, the layout of the environment is generally unknown. Hence, there is a need to explore the environment in order to effectively perform search and rescue. Exploration of unknown environments using a single robot is a well studied problem. We present an algorithm to perform the task with a team of p robots for the specific case of orthogonal polygons, i.e. polygonal environments where each side is aligned with either the X or the Y axis. The 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 optimal offline algorithm. We then optimize the information gain of the path followed by the robots by allowing local detours in order to decrease the entropy in the map.

This work is supported by NSF through grant number 1566247.

To my family for their unwavering *love* and *support*.

Acknowledgments

First and foremost, I would like to thank my advisor Dr.Pratap Tokekar for giving me the opportunity to work with him in the Robotics Automation and Autonomous Systems (RAAS) laboratory at Virginia Tech. His *Advanced Topics in Robotics* class kick started my journey in robotics and was one of the best classes here. He has been a constant source of inspiration and his knowledge in various areas is awe-inspiring. Honestly, He is the best advisor anyone can ever hope for and I hope to work with him in the future.

I would also like to thank Dr.Sharath Raghvendra and Dr.Daniel Stilwell for being a part of my comittee. Dr.Raghvendra's *Theory of Algorithms* class was one of the best classes I ever attended and it helped deepen my understanding of algorithms. I would also like to thank Dr.Devi Parikh as well for her lectures in the computer vision, I learnt a lot during the course and it was a lot of fun working on the assignments and the class project.

Special thanks to my lab-mate and co-author Kevin Yu for helping me with the real world experiments and spending many sleepless nights in the corridors of the whittemore hall running behind the robot. The others, Ashish, Zhongshun, Nahush, Yoon and Lifeng made the lab a great place to work in.

I thank my friend Shriya Shah for being a guide and making amazing food for us. Also, my friends Amith, Arpit, Nisheeth and Jai for making my journey as a graduate student less stressful.

My heartfelt thanks to my parents, Prem and Shanthi (*Love and peace*), and my brother, Ranjith, for always believing in me and pushing me to chase my dreams. They have been extremely encouraging throughout my life and I owe everything that I have ever achieved to them.

Lastly, I would like to extend my sincere thanks to my better half, Nayanika Sanga, for enduring this journey with me. I know that these two years have been a lot more difficult for her than me. She has been a constant source of support and she always believed in me, even when I didn't. Thank you and I can't wait to spend the rest of my life with you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Exploration of unknown environments using a single robot has been a well studied problem [46, 31]. The task can be performed 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. The main contributions of this thesis is an algorithm for multi-robot exploration of unknown environments with strong theoretical performance guarantees.

There have been two types of approaches towards solving the exploration problem: geometric and information-theoretic. In geometric approaches (e.g., [6]), it is typical to assume that the robots have perfect 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., [10]) explicitly take into account practical constraints such as noisy sensors and complex environments. However, these approaches are often greedy (e.g., frontier-based [55]) and typically do not yield any guarantees on the total time taken. In this dissertation we investigate the challenges in combining information-theoretic algorithms with geometric exploration algorithm while preserving guarantees on exploration time.

We use competitive analysis [38] 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

$$Competitive\ ratio = \max_{i\ \in\ input} \frac{Cost\ of\ online\ algorithm(i)}{Cost\ of\ optimal\ offline\ algorithm(i)}$$

The optimal offline algorithm corresponds to the case when the input (i.e., map of the environment) itself is known. The goal is to find online algorithms with small, constant competitive ratios. That is, algorithms whose online performance is comparable to algorithms who know the input a priori. We present a algorithm for multi-robot exploration with a constant competitive ratio for exploring with p robots, when p is fixed in orthogonal polygons.[1]

1.2 Background and 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.

1.2.1 Polygon Exploration

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 [42], watchman route problem [12], and target search [22] and shortest path planning [44] in unknown environments.

Using a fixed set of positions for guarding a known n-sided polygonal region, i.e., a set of points from which the entire polygon is visible, is known as the classical art gallery problem. Chvatal [14] and Fisk [21] 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 [48] 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 [12] showed that the watchman route can be found in polynomial time in a simple orthogonal polygon. Wang et al. [53] showed that the watchman route problem is for general environments is NP-hard and presented a $\mathcal{O}(\text{polylog} n)$ approximation for the restricted case when each viewpoint is required to see a complete polygon edge.

Exploring an unknown polygon is the online watchman route problem. Bhattacharya et al. [6] and Ghosh et al. [24] approached the exploration problem with discrete vision, i.e., they assume that the robot does not have continuous visibility and has to stop at different scan points in order to sense the 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 limited range of visibility, they give an algorithm where

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

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 and R is the number of reflex vertices in the polygon.

Albers et al. [1] assume that the environment is modeled by a directed, strongly connected graph and give a $d^{\mathcal{O} \log d} m$ competitive algorithm where where m is the number edges in the graph and d is the minimum number of edges that have to be added to make the graph Eulerian. The robot's task is to visit all nodes and edges of the graph using the minimum number R of edge traversals.For a simple polygon, Hoffmann et al. [26] presented an algorithm which achieves a competitive ratio of 26.5. For the special case of an orthogonal polygon, Deng et al. [17] presented a $\sqrt{2}$ competitive exploration strategy with one robot. We show how to extend the single robot exploration algorithm by Deng et al. [17] to the case of p robots. The resulting algorithm has a competitive ratio that is a function of p.

1.2.2 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 beforehand 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 [36]. For the Euclidean version of the problem, there exist polynomial time approximation schemes [36, 4], 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. [23] 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. [18] 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. [25] 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. [19] 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 $\mathcal{O}(\log^2 d)$ [43]. 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. Our algorithm yields a competitive ratio of $\frac{2(2\sqrt{2}p+\log p)}{1+\log p}$ where p is the number of robots.

1.2.3 Information-Theoretic Exploration

Geometric and graph-based approaches typically assume perfect sensing with no noise. In practice, however, measurements are not perfect and we have to account for measurement noise when exploring the environment. Such exploration strategies can be broadly classified into frontier-based and information-theoretic strategies [31].

Information-Theoretic exploration strategies typically use an occupancy grid to represent the maps generated from noisy and uncertain sensor measurement [20]. An occupancy grid is a uniformly-spaced grid, with a binary random variable (per grid cell) representing the probability of the cell being occupied by an obstacle. There are many existing occupancy grid mapping techniques like gmapping [50], octomap [29], RTAB-Map [35] and ORGB-slam [40]. Frontier-based exploration strategies are largely greedy in nature and drive the robots to the boundary between known and unknown spaces in the map. In occupancy grids frontiers are cells determined to be free (Probability of occupancy close to zero) which are next to grid cells that have not been observed (Probability of occupancy is set to be 0.5). Variants of this strategy have been used to perform exploration of unknown 2D [55, 8, 49] and 2.5D environments [9]. We refer the reader to the comprehensive study by Holz et al. [28] for further details. Information-theoretic strategies seek to optimize some information measure

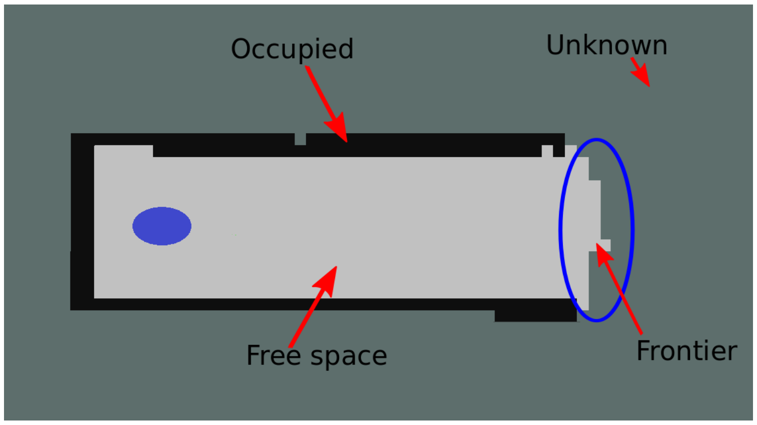

Figure 1.1: Frontier is the boundary between known free space and unknown space.

such as entropy (uncertainty) [54, 37] in the environment, or mutual information [3] while exploring the environment. Mutual information [15] between two random variables X and Y is given by,

$$I(X;Y) = \sum_x \sum_y f(x,y) \ \log \frac{f(x,y)}{f_1(x)f_2(y)}$$

Mutual information in occupancy grid, predicts how much future measurements will decrease the robots uncertainty associated with all grid cells. Julian et al. [32] studied the computation of mutual information for range based sensors. While these algorithms produce better maps, the planner is typically a one-step greedy algorithm or finite-horizon planners that cannot give global guarantees on the total distance traveled.

In order to overcome this, a better approach may be to combine a global planner along with a local planner. Davis et al. [16] proposed one such algorithm where the goal was to plan a path from a start state to a goal state while the coverage of a user-specified region while minimizing the control costs of the robot and the probability of collision with the environment. Bai et al. [5] have used used an approach to predict mutual information using Bayesian optimization in which the long-term goal is to reduce entropy throughout the robot's environment map, and the short-term goal is to perform the sensing action in each iteration that will maximize mutual information. Choudhury et al. [13] have showed that supervised learning could be used to predict informative actions without evaluating the expected mutual information exhaustively for every possible action. Charrow et al. [10] attempted to resolve this with a heuristic that uses a global planner for a single robot to determine trajectories that maximizes the information-theoretic objective whilst employing a gradient-based trajectory optimization technique to locally refine the chosen trajectory such that the mutual information objective is maximized. We build on this work and present a two-level planner that maximizes information locally while giving strong global performance guarantees on the path length followed by the robots.

1.3 Contributions of this Thesis

We focus on the case of exploring unknown orthogonal polygons without any holes. Deng et al. [17] showed that there is an algorithm with a constant competitive ratio for exploring orthogonal 2-dimensional polygons with a single robot.We present an algorithm with constant competitive ratio for exploring with p robots, when p is fixed (Chapter 2).

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 (Chapter 2). Thirdly, we extend this algorithm in order to improve the quality of the resulting map. We add a local planner to our algorithm that optimizes the information gain of the path taken by the robots while traversing in order to reduce the overall uncertainty in the map. We evaluate our algorithm through simulations and experiments on a mobile robot (Chapters 3 and 4).

Chapter 2

Tree Exploration with Trajectory Optimization

In this chapter, 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. [17] for exploring an orthogonal polygon with a single robot and extends it to the case of multiple robots using the graph exploration strategy from [25]. Our main insight is to show that the path followed by the robot using the algorithm in [17] 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 [25] 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).

2.1 Problem Formulation and Preliminaries

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 2.1) 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

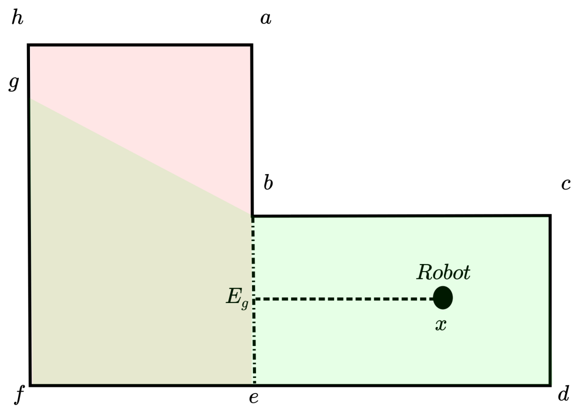

Figure 2.1: Vertex b blocks the visibility of the robot and is known as a *blocking vertex.* The robot has to cross the segment $\overline{be}$ (known as the *extension*) to increase the visible boundary of the polygon P. The *extension* divides P into two sub-polygons. The sub-polygon not containing the robot (overlap of red and green areas) is known as the *foreign polygon* w.r.t. b and is denoted as $FP(b)$.

boundary of P where as others are chords in the interior of P (e.g., segment $\overline{gb}$ in Figure 2.1). 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 2.1 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 E divides P into two sub-polygons. The one which contains the robot is known as the *home polygon* of the robot with respect to E. The other sub-polygon is known as the *foreign polygon* of the robot with respect to E and is denoted as $FP(E)$.

For two extensions E_1 and E_2, there may be no way to visit E_1 without crossing E_2. If so, we can ignore E2, since it is automatically visited if we visit E_1. More formally, E_1 is said to dominate E_2 if E_1 is totally contained $FP(E_2)$. A non-dominated extension is called a critical extension.

2.2 Algorithm

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

```
Function explore()
    Data: Cluster of robots, $C$, located at some node, $A$, in the tree.
    if $A$ is marked under-exploration then
        $\mathcal{A} \leftarrow$ children of $A$ not marked as explored;
        if $\mathcal{A} == \emptyset$ then
            if $A ==$ root of the tree then
                Terminate exploration;
            else
                goto($C$,parent($A$);
            end
        else
            Divide $C$ equally among $\mathcal{A}$;
            When any cluster reaches a child of $A$, call explore;
        end
    else
        Mark $A$ as under-exploration;
        if blocking vertices detected from $A$ then
            Sort in clockwise direction and add as children of $A$;
            for $p$ and $q$ are distinct blocking vertices do
                if $FP(p) \subseteq FP(q)$ then
                    add $p$ as child of $q$;
                else
                    if $FP(p)$ and $FP(q)$ intersect then
                        add the vertex that appears first in the clockwise order as the parent
                          of the vertex that appears later;
                    end
                end
            end
            Divide $C$ equally among children of $A$;
            When any cluster reaches a child of $A$, call explore;
        else
            Mark $A$ as explored;
            goto($C$,parent($A$);
        end
    end
```

Algorithm 1: Multi-Robot Exploration Subroutine

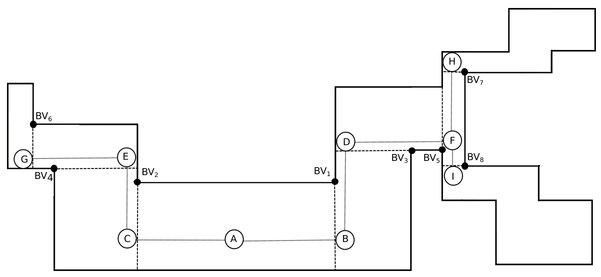

Figure 2.2: An intermediate stage of exploration of a polygon by a team of 4 robots. Two robots located at the location G and the other two are located at F.

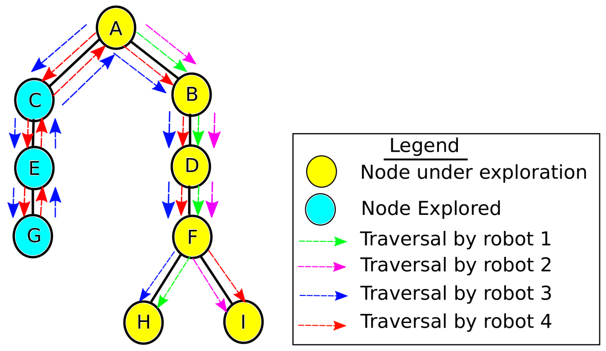

Figure 2.3: The tree corresponding to the stage of exploration shown in Figure 2.2.

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 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 2.2. 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 2.3. 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*.

2.3 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. The graph created by the algorithm is a tree by construction because once a node is added to the graph it is not added to the graph again.

When $p = 1$, the proposed algorithm is the same as the one given by Deng et al. [17] 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 [17] for a single robot to explore P. We have $C_{\text{RECT}} \leq \sqrt{2}C_{\text{OPT}}$ from Theorem 3 in [17] 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 $C_{\text{ALG}}/C_{\text{OPT}}$. We will show this by relating both quantities with C^p_{TREE} which is the sum of the lengths of edges in the tree created by p robots.

Lemma 1. $C^1_{TREE} \leq C_{RECT} \leq \sqrt{2}C^1_{OPT}$.

Proof. The inequality $C_{\text{RECT}} \leq \sqrt{2}C^1_{\text{OPT}}$ holds from Theorem 3 of [17] for simple rectilinear (orthogonal) polygons. The inequality $C^1_{\text{TREE}} \leq C_{\text{RECT}}$ holds because any robot will have to back track on its path to reach previously unexplored areas and return back to the root. □

Lemma 2. *If p robots are used to explore P and C^p_{OPT} is the cost of the optimal offline algorithm, then $C^1_{TREE} \leq \sqrt{2}C^1_{OPT} \leq \sqrt{2}pC^p_{OPT}$.*

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^1_{OPT} is the optimal cost for a single robot's tour, we have $C^1_{\text{OPT}} \leq pC^p_{\text{OPT}}$. The other inequality follows from the previous lemma. □

Lemma 3. *The single robot visits the critical extensions of the sides that appear in the clockwise order on the boundary of the polygon. In multi-robot exploration, the total order is not preserved but the partial order is preserved.*

Proof. For the single robot case, this holds from *Proposition 4* of [17]. For the multirobot case it holds by construction of the exploration algorithm (*Line 17*). □

Lemma 4. *If C^p_{TREE} is the cost of the tree created by p robots, then $C^p_{TREE} \leq 2C^1_{TREE}$.*

Proof. Let the single robot following the strategy from [17] visit the critical extensions in the clockwise order in which the sides associated with the extensions appear along the boundary of the polygon, $S_1, S_2, ..., S_{\text{i}}, S_{\text{j}}, S_{\text{k}}, ..., S_{\text{m}}$ starting from root location r. Let $C(a, b)$ denote the cost of traversing from point a to point b following the rectilinear strategy. Therefore,

$$C^1_{\text{TREE}} = C(r, S_1) + C(S_1, S_2) + ... + C(S_{\text{i}}, S_{\text{j}}) + C(S_{\text{j}}, S_{\text{k}}) + C(S_{\text{m-1}}, S_{\text{m}}) \tag{2.1}$$

From Lemma 3, we know that each of the p robots will visit the sides in a partial order. We show that the cost of the tree constructed by p robots is upper bounded by the cost of the partial order tours. Next, we show that the cost of the partial order tours is no more than twice the cost of the total order tour. We will show this for the case of two partial order tours. The same argument can be applied iteratively to convert all partial order tours to a total order tour.

Now, let us assume that we have two robots in the exploration team and the first robot visits the extensions in the following order $S_1, S_2, ..., S_{\text{i}}, S_{\text{k}}, ..., S_{\text{m}}$ and the second robot visits the extension S_{j}. The cost of the combined tour is given by

$$C^p_{\text{TREE}} = C(r, S_1) + C(S_1, S_2) + ... + C(S_{\text{i}}, S_{\text{k}}) + C(S_{\text{m-1}}, S_{\text{m}}) + C(r, S_{\text{j}}) \tag{2.2}$$

By triangle inequality, we can see that,

$$C(r, S_{\text{j}}) \leq C(r, S_1) + C(S_1, S_2) + ... + C(S_{\text{i-1}}, S_{\text{i}}) + C(S_{\text{i}}, S_{\text{j}}) \tag{2.3}$$

Also,

$$C(S_{\text{i}}, S_{\text{k}}) \leq C(S_{\text{i}}, S_{\text{j}}) + C(S_{\text{j}}, S_{\text{k}}) \tag{2.4}$$

Therefore from equation 2.3 and equation 2.4,

$$\begin{aligned} C(r, S_1) + C(S_1, S_2) + ... + C(S_{\text{i}}, S_{\text{k}}) + C(S_{\text{m-1}}, S_{\text{m}}) + C(r, S_{\text{j}}) \leq{} & C(r, S_1) + C(S_1, S_2) + \\ & ... + C(S_{\text{i-1}}, S_{\text{i}}) \\ & + (C(S_{\text{i}}, S_{\text{j}}) + C(S_{\text{j}}, S_{\text{k}})) \\ & + (C(r, S_1) + C(S_1, S_2) + ... + \\ & C(S_{\text{i-1}}, S_{\text{i}}) + C(S_{\text{i}}, S_{\text{j}})) \end{aligned} \tag{2.5}$$

From equation 2.2,

$$C^p_{\text{TREE}} \leq 2(C(r, S_1) + C(S_1, S_2) + ... + C(S_{\text{i}}, S_{\text{j}}) + C(S_{\text{j}}, S_{\text{k}}) + ...C(S_{\text{m-1}}, S_{\text{m}})) \tag{2.6}$$

From equation 2.1,

$$C^p_{\text{TREE}} \leq 2C^1_{\text{TREE}} \tag{2.7}$$

Each such addition adds a factor of of 2 and hence we can do this for multiple tours without increasing the cost of the tour by more than a factor of 2.

□

Theorem 1. *If $C_{EXPLORE}$ 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,*

$$\frac{C_{EXPLORE}}{C^p_{OPT}} \leq \frac{2(2\sqrt{2}p + \log p)}{1 + \log p} \tag{2.8}$$

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

$$C_{\text{EXPLORE}} \leq \frac{2(C^p_{\text{TREE}} + d_{\max} \log p)}{1 + \log p}, \tag{2.9}$$

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 [25]. 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 $d_{\max} \leq C^p_{\text{OPT}}$.

From Lemma 2 and 4, we have:

$$C_{\text{EXPLORE}} \leq \frac{2(2\sqrt{2}pC^p_{\text{OPT}} + C^p_{\text{OPT}}\log p)}{1+\log p}$$

which yields

$$\frac{C_{\text{EXPLORE}}}{C^p_{\text{OPT}}} \leq \frac{2(2\sqrt{2}p + \log p)}{1+\log p} \tag{2.10}$$

□

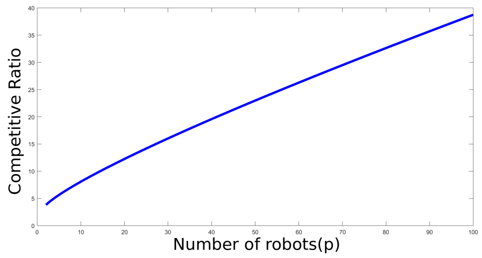

Figure 2.4: Plot of the growth of competitive ratio as a function of the number of robots.

Thus, we show that the competitive ratio is bounded as a function of p. Figure 2.4 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.

2.4 Incorporating Information-Theoretic Planning

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.

2.4.1 Finding Blocking Vertices in Occupancy Maps

The assumption we make 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 2.5-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 beams from the laser as shown in Figure 2.5-left. The beams 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.

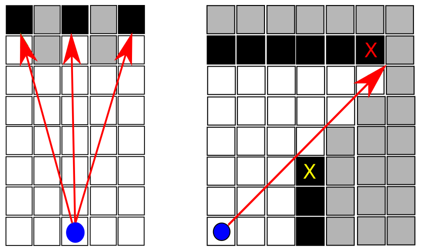

Figure 2.5: **(Left)** Finite resolution of the laser leads to gaps in observations. **(Right)** A blocking vertex (marked with yellow 'X') has four distinctive neighbors: an occupied cell, an unknown cell, a free cell which is a frontier, one which is not.

Consider the robot (and the laser) located at the blue circle in Figure 2.5-right. The red ray represents one of the laser beams. 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 2.5-right, the cell marked with the yellow *'X'* is detected as a blocking vertex.

2.4.2 Information-Theoretic Subroutine

In the analysis for Algorithm 1, we assumed that we follow the shortest paths between A and B when executing the `goto(A,B)` subroutine. Instead, we can add local detours that will increase the info gain in order to improve the quality of the map. In order to maintain a constant competitive ratio, the robot is given a certain budget. This is known as the orienteering problem, i.e, given a start position, a goal position and a graph where each node has an associated reward, the orienteering problem seeks to maximize a reward function constrained to a certain budget [7]. Our objective is to minimize the uncertainty in the map, hence mutual information was determined to be the most suitable reward function. Since mutual information is known to be a submodular function. A submodular function is a set function whose value, informally, has the property that the difference in the incremental value of the function that a single element makes when added to an input set decreases as the size of the input set increases. We use the recursive greedy algorithm for submodular reward functions described in [11] while moving between nodes in the tree.

The orienteering subroutine takes as input: input graph, G, the start vertex, s, the goal vertex, t, budget, B, visited set of nodes, X and returns a path that maximizes a submodular reward function subject to the budget. The input graph is generated by imposing a grid (add points above and below the path) on the shortest path between s and t as shown in Figure 2.6.

Orienteering is known to be NP-complete, and so is submodular orienteering. Chekuri and Pal [11] give a quasi-polynomial time recursive greedy algorithm that yields an $\mathcal{O}(\log\ OPT)$ approximation for this problem. We modify the subroutine in order to save computation time by restricting the paths to be strictly moving forward, i.e, the path cannot have edges moving back towards s. This reduces the computation time by an order of magnitude. This is a heuristic and therefore the approximation ratio may not necessarily hold in this case.

Theorem 2. *Let B be the assigned budget to the algorithm, i.e., the robot is allowed to take a path of length B times the shortest distance between the nodes. Let C_{ALG} be the cost of exploring the the polygon using the proposed algorithm along with the information theoretic subroutine. We have,*

$$\frac{C_{ALG}}{C_{OPT}} \leq \frac{2B(2\sqrt{2}p + \log p)}{1 + \log p} \tag{2.11}$$

Proof. From Theorem 1, we have:

$$\frac{C_{\text{EXPLORE}}}{C_{\text{OPT}}} \leq \frac{2(2\sqrt{2}p + \log p)}{1 + \log p} \tag{2.12}$$

Each $s - t$ path between nodes is allotted a budget B, hence the total cost of our algorithm C_{EXPLORE} is multiplied by a factor of B. Hence the total cost of the algorithm with the information theoretic subroutine is given by,

$$C_{\text{ALG}} = BC_{\text{EXPLORE}} \tag{2.13}$$

Hence,we have,

$$\frac{C_{\text{ALG}}}{C_{\text{OPT}}} \leq \frac{2B(2\sqrt{2}p + \log p)}{1 + \log p} \tag{2.14}$$

□

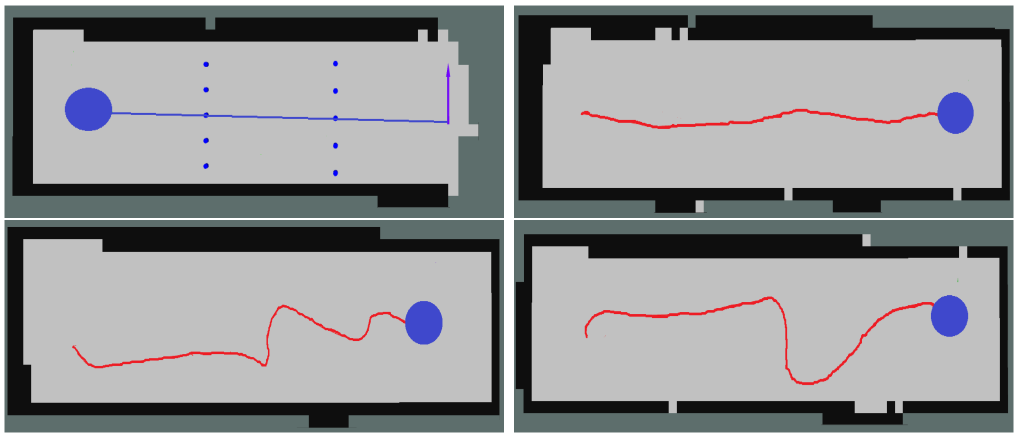

Figure 2.6: The first figure (Top-Left) shows the input graph imposed on the shortest path (blue line) between s and t. The other figures show the variation of path taken with change in budget.

2.4.3 General Environments

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 chapter, we evaluate the empirical performance of our algorithm through simulations in such scenarios.

Chapter 3

Simulation Results

We evaluate our algorithms via simulations and real world experiments. ROS [45] was used extensively for our experiments. For the simulations, we used the gazebo simulator [33] which is a physics-based simulator.

3.1 Simulations with `goto` as the shortest path

We first ran the experiments with the `goto` function as the shortest distance between s and t. The five gazebo simulation environments used (Figure 3.1) are not all orthogonal and simply-connected – assumptions required for the analysis. We ran experiments with varying number of robots in all the environments and evaluated the performance of our algorithm. [1]

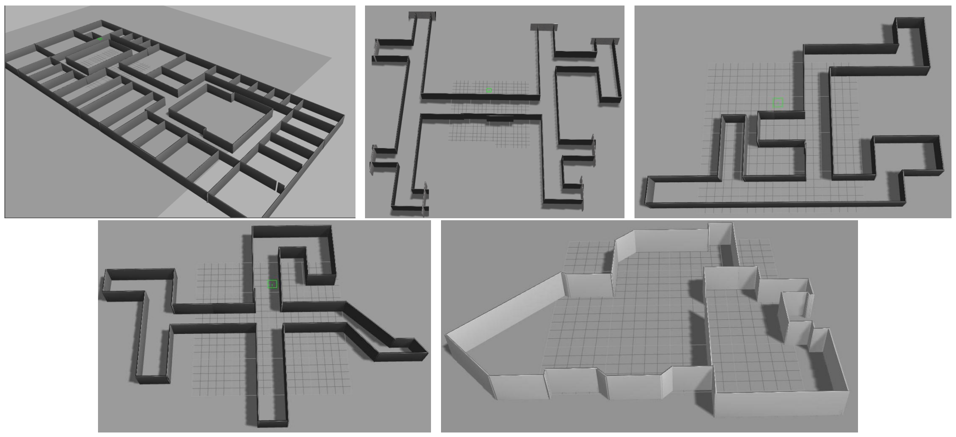

Figure 3.1: Simulation environments 1 – 5 from left to right, top to bottom.

[1]Our implementation is available online at `https://github.com/raaslab/Exploration`.

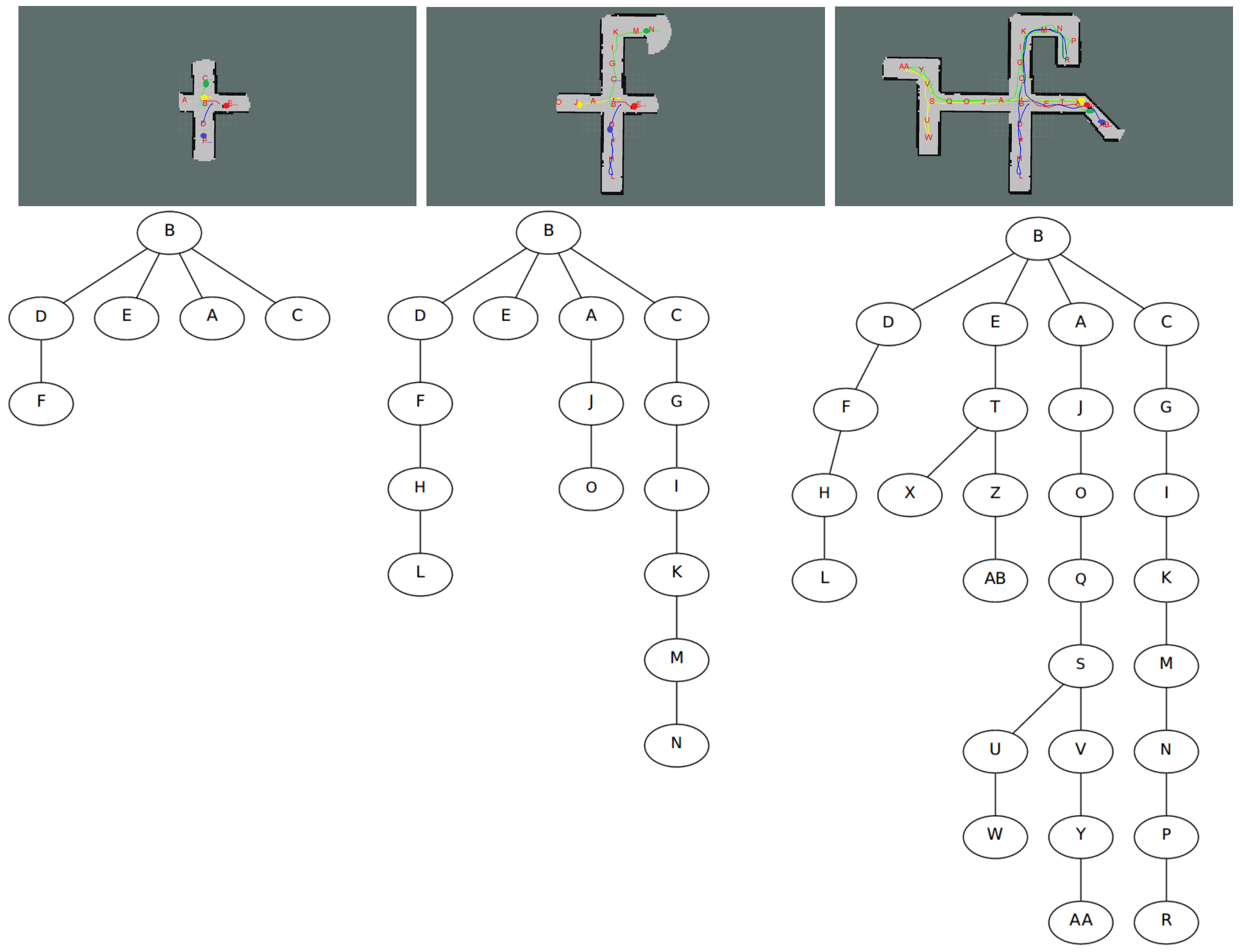

Figure 3.2: Various stages while exploring environment 3 using four robots. The map explored and built along with the corresponding trees are shown.

Figure 3.2 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 3.3. Table 3.1 shows the maximum distance traveled by a robot (in meters) during exploration for all the environments.

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 3.3). 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.

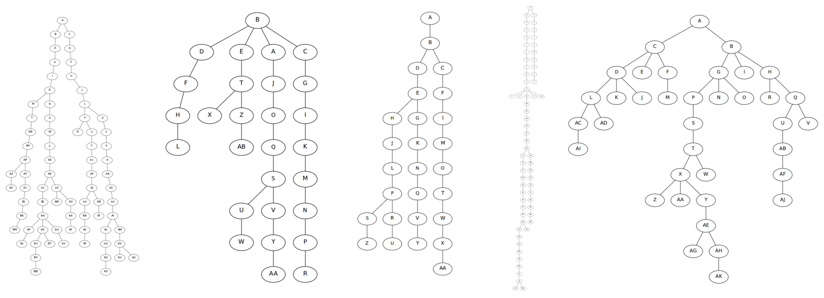
Figure 3.3: Final tree produced after exploring the five environments in Figure 3.1 with four robots. The maximum distance traveled by a robot during exploration is given in Table 3.1.

	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 3.1: Maximum distance traveled by a robot to explore environments in Figure. 3.1.

3.2 Simulations with `goto` as the submodular orienteering subroutine

We replace the `goto` function and run the experiments in environment 1 and varying the budget and number of robots. Table 3.2 shows the effect of varying the budget allowed to different teams of robots. We observe that the entropy in the environment drops with increase in budget, which is expected. We also observe that increase in budget increases the total exploration time as well, this is due to the fact that that recursive greedy algorithm iterates over all possible budgets and hence, the per step computation time increases with increase in budget. Per step computation here refers to a single call to goto subroutine. Table 3.3 shows the increase in average (over multiple steps of a single run) per step computation time with increase in budget for the case of a single robot. The graph is a 4-connected grid graph and is restricted to have a maximum of 20 nodes, irrespective of the distance between s and t. We use the approach of Charrow et al. [10] to compute the mutual information at each location in the graph.

Figure 3.4 shows the final map after exploration using two robots with budgets $1d$ and $2d$. We observe that the robots take straighter paths to the goal when the budget is $1d$ and bend their paths when the budget is increased in order to maximize the mutual information gain of the path.

	Budget	**Entropy**	**Total Time**
1 Robot	$1d$	24713	1131
1 Robot	$2d$	24479	2719
1 Robot	$4d$	24131	6187
2 Robots	$1d$	24400	949
2 Robots	$2d$	24267	3406
2 Robots	$4d$	24039	6584

Table 3.2: Effect of budget while exploring first environment in Figure 3.1, where d is the shortest distance between every start (s) and goal (t).

Budget	**Computation Time(s)**
$1d$	3.41
$2d$	7.12
$4d$	19.38

Table 3.3: Average per step computation time for varying budget, where d is the shortest distance between every start (s) and goal (t).

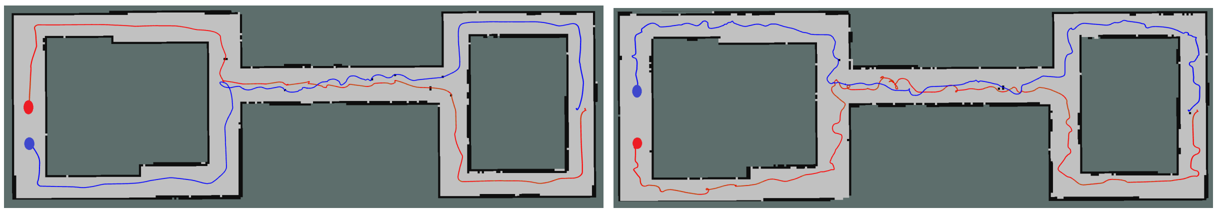

Figure 3.4: **(Left)** Final map produced and path taken by two robots with an orienteering budget of $1d$. **(Right)** Final map produced and path taken by two robots with an orienteering budget of $2d$.

Chapter 4

Hardware Experiments

4.1 System Description

In this chapter, we first describe the ground robot system for conducting the experiments. We the present the results of our experiments using the ground robots on the sixth floor of Whittemore hall and RAAS Lab.

4.1.1 Robot Localization

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 4.1). We used the `RosAria` package [47] for interfacing with the robot. 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` [35, 34] and the localization was carried out using the `amcl` [2] package from ROS. `amcl` is a probabilistic localization system for a robot moving in 2D. It implements the adaptive (or KLD-sampling) Monte Carlo localization approach [51], which uses a particle filter to track the pose of a robot against a known map. Figure 4.2 shows the `rqt_graph` showing the communication between various nodes used for localization, `amcl` localizes the robot by using the pre-built map from `map_server`, the laser scan from `hokuyo_node` [27] and the odometry information from `RosAria`. `amcl` also requires the transformation between the laser and the robot's base, which is provided by `base_link_to_laser` node. We use a joystick for manual operation of the robot while building the map offline. The `joy` [30] node publishes a `joy` message, which contains the current state of each one of the joystick's buttons and axes. The `joy` messages are then converted into command velocity messages to control the robot. 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 like `gmapping` [50].

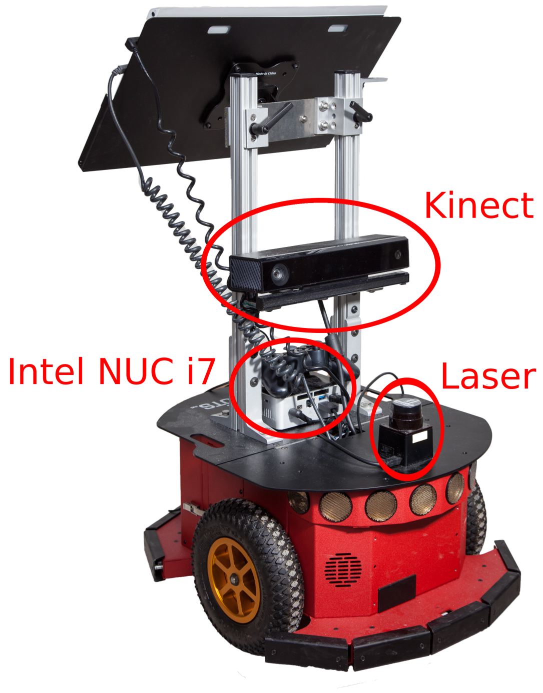

Figure 4.1: Pioneer P3-DX with the onboard Intel NUC i7, Hokuyo laser and Kinect2 sensor used for the experiments. The 2D Hokuyo laser was used for robot localization and the Kinect2 sensor was used for mapping during exploration.

4.1.2 Online Map Building

The robots generated a new map during the exploration process using `octomapping` [29] with the Kinect2 sensor. The OctoMap library implements a 3D occupancy grid mapping approach. The map implementation is based on an octree data structure. 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. A single NUC is not enough for processing the Kinect data and running the exploration algorithm. Hence, the slave i7 Intel NUC was dedicated to processing the Kinect2 data and the master i7 Intel NUC was dedicated to running the localization, mapping and exploration algorithm.

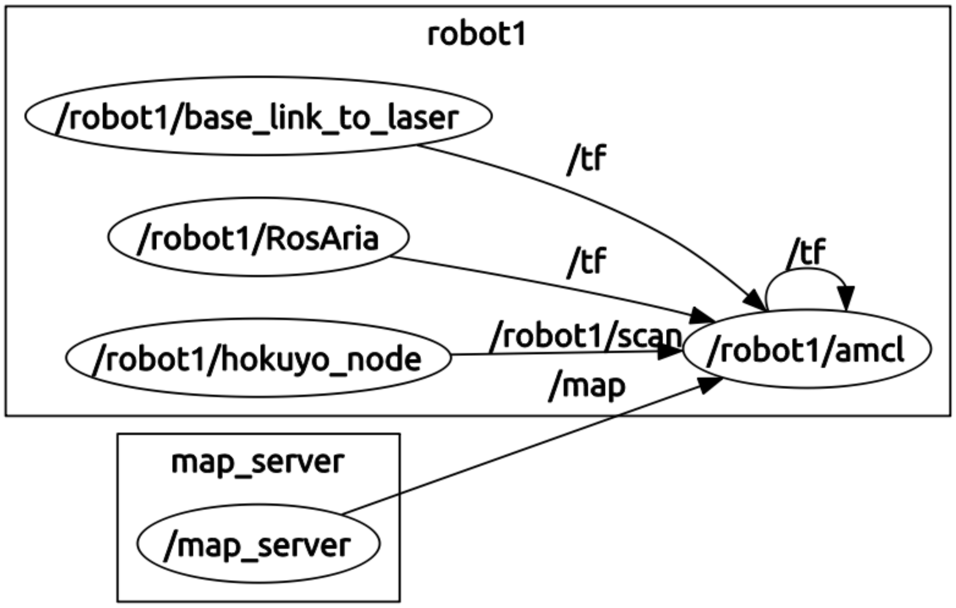

Figure 4.2: rqt_graph showing communication between nodes.

4.1.3 Multi-Robot Communication

For multi-robot experiments, we setup global communication between the robots and a central computer. In order to achieve this, they are connected to the same wireless network setup using bullet m3 antennas [52] (Figure4.4). The antennas are setup in a mesh topology. We use the `multimaster_fkie` [39] package for setting up multiple masters on the intel NUCs. `multimaster_fkie` is a metapackage to combine the nodes required to establish and manage a multimaster network. We use `occupancy_grid_utils` [41] to combine occupancy grids and run the exploration algorithm.

4.2 Experiments

The sensor on the robot is more noisy than that in simulations. We modified the goal selection by first checking whether a frontier goal (or extension goal) is reachable from the robots current position. Figure 4.5 shows the results of an exploration experiment with a single robot in a corridor environment along with the path followed by the robot.[1]

Table 4.1 shows the effect of budget on the entropy in the map using the exploration strategy with the information theoretic subroutine. We observe that as with the simulations, increase in budget decreases the overall entropy in the environment but increases the total exploration

[1]A video of the system in operation is available online at `https://github.com/raaslab/Exploration.`

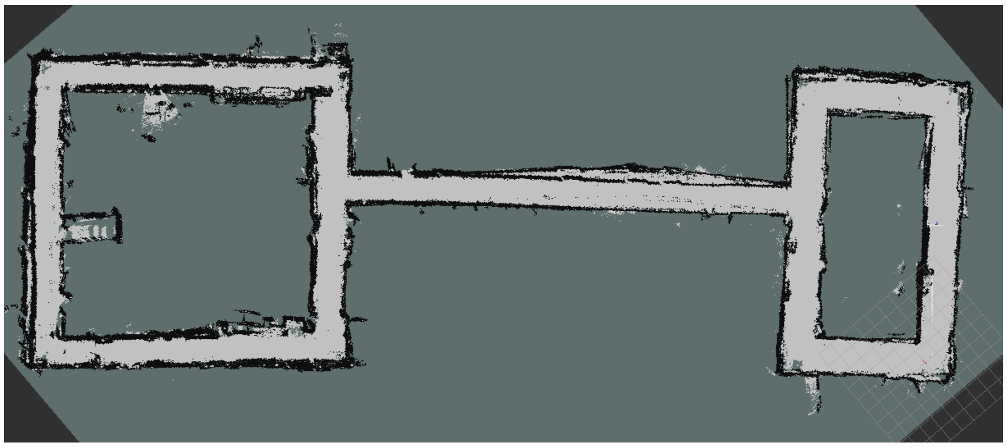

Figure 4.3: Map of sixth floor of Whittemore hall built using rtabmap_ros package.

Figure 4.4: Bullet M3 antennas were used to setup global communication.

time.

Figure 4.7 shows the result of exploration of part of Whittemore hall and the path followed by the two robots. We were unable to conduct experiments with multiple robots while incorporating the information theoretic subroutine because we couldn't combine the octomaps produced by the two robots.

	Budget	**Entropy**	**Total Time(s)**
1 Robot	$1d$	78493	1516
1 Robot	$2d$	78128	2453
1 Robot	$4d$	77917	5427

Table 4.1: Effect of budget while exploring the corridor shown in Figure 4.3 using the real robot, where d is the shortest distance between every start (s) and goal (t).

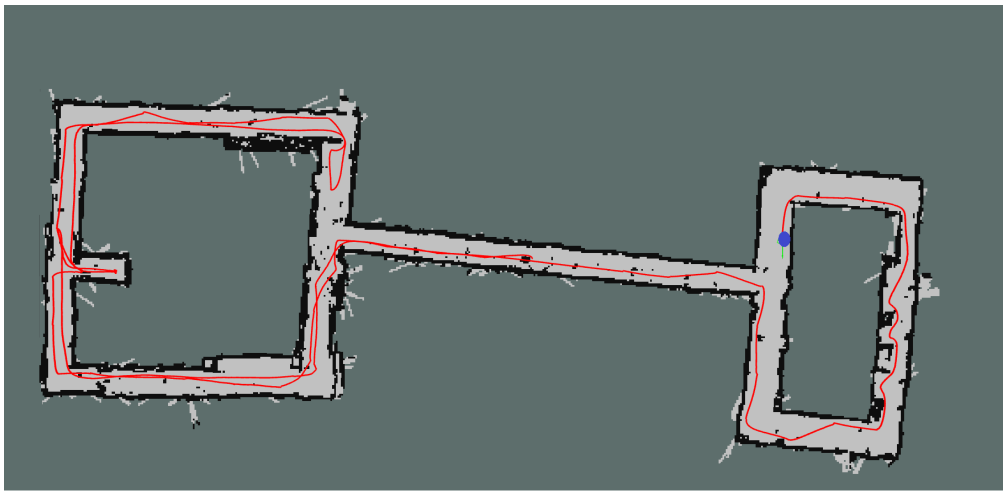

Figure 4.5: Map of sixth floor of Whittemore hall built and the path taken by the robot after exploration of the environment.

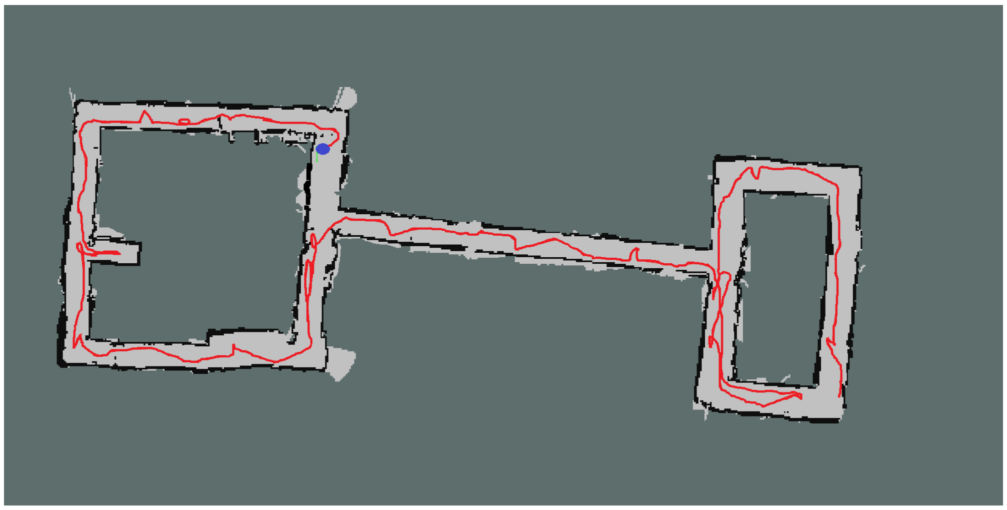

Figure 4.6: Map of sixth floor of Whittemore hall built and the path taken by the robot after exploration of the environment with a budget of $2d$.

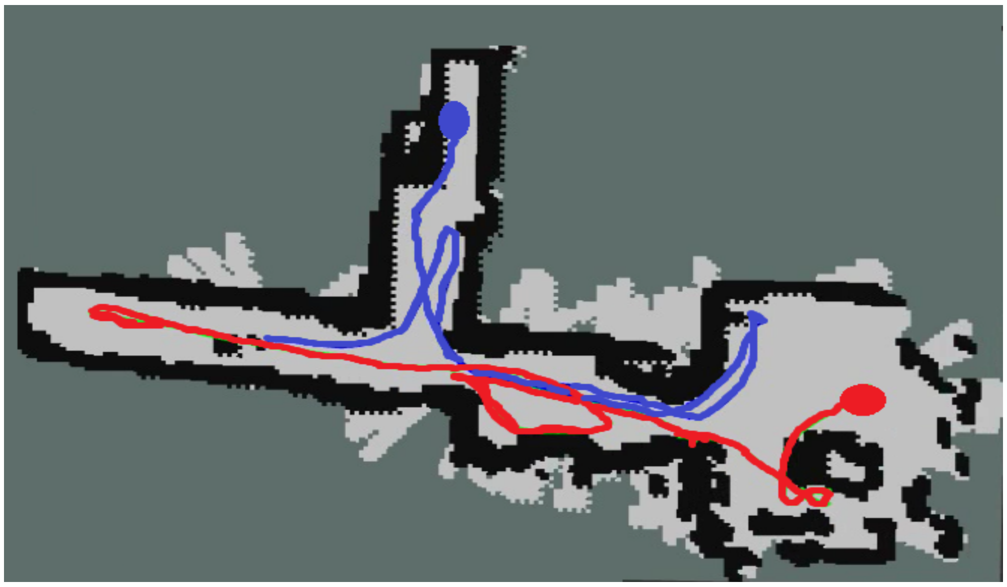
Figure 4.7: Map of RAAS Lab and part of sixth floor of Whittemore hall built and the path taken by 2 robots.

Chapter 5

Conclusion and Future Work

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. Furthermore, to improve the quality of the resulting map, while traveling to goal positions, the robots were alloted an extra budget and we used an existing recursive greedy solution to solve for sub-modular orienteering to maximize the mutual information gain locally. We verified the behavior of our algorithm through simulations and experiments with a single robot.

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 [17] 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.

Bibliography

[1] Susanne Albers and Monika R Henzinger. “Exploring unknown environments”. In: *SIAM Journal on Computing* 29.4 (2000), pp. 1164–1188.

[2] *AMCL ROS Packagel*. `http://wiki.ros.org/amcl`. Accessed: 2016-10-30.

[3] Francesco Amigoni and Vincenzo Caglioti. “An information-based exploration strategy for environment mapping with mobile robots”. In: *Robotics and Autonomous Systems* 58.5 (2010), pp. 684–699.

[4] Sanjeev Arora. “Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems”. In: *Journal of the ACM (JACM)* 45.5 (1998), pp. 753–782.

[5] Shi Bai et al. “Information-theoretic exploration with Bayesian optimization”. In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 1816–1822.

[6] Amitava Bhattacharya, Subir Kumar Ghosh, and Sudeep Sarkar. “Exploring an unknown polygonal environment with bounded visibility”. In: *International Conference on Computational Science*. Springer. 2001, pp. 640–648.

[7] Avrim Blum et al. “Approximation algorithms for orienteering and discounted-reward TSP”. In: *SIAM Journal on Computing* 37.2 (2007), pp. 653–670.

[8] Wolfram Burgard et al. “Coordinated multi-robot exploration”. In: *IEEE Transactions on robotics* 21.3 (2005), pp. 376–386.

[9] Kyle Cesare et al. “Multi-UAV exploration with limited communication and battery”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 2230–2235.

[10] Benjamin Charrow et al. “Information-theoretic planning with trajectory optimization for dense 3d mapping”. In: *Proceedings of Robotics: Science and Systems*. 2015.

[11] Chandra Chekuri and Martin Pal. “A recursive greedy algorithm for walks in directed graphs”. In: *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*. IEEE. 2005, pp. 245–253.

[12] Wei-pang Chin and Simeon Ntafos. “Optimum watchman routes”. In: *Information Processing Letters* 28.1 (1988), pp. 39–44.

[13] Sanjiban Choudhury et al. "Learning to Gather Information via Imitation". In: *arXiv preprint arXiv:1611.04180* (2016).

[14] Vasek Chvatal. "A combinatorial theorem in plane geometry". In: *Journal of Combinatorial Theory, Series B* 18.1 (1975), pp. 39–41.

[15] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[16] Bobby Davis, Ioannis Karamouzas, and Stephen J Guy. "C-opt: Coverage-aware trajectory optimization under uncertainty". In: *IEEE Robotics and Automation Letters* 1.2 (2016), pp. 1020–1027.

[17] Xiaotie Deng, Tiko Kameda, and Christos Papadimitriou. "How to learn an unknown environment. I: the rectilinear case". In: *Journal of the ACM (JACM)* 45.2 (1998), pp. 215–245.

[18] Miroslaw Dynia, Jakub ŁopuszaŃski, and Christian Schindelhauer. "Why robots need maps". In: *International Colloquium on Structural Information and Communication Complexity*. Springer. 2007, pp. 41–50.

[19] Miroslaw Dynia et al. "Smart robot teams exploring sparse trees". In: *International Symposium on Mathematical Foundations of Computer Science*. Springer. 2006, pp. 327–338.

[20] Alberto Elfes. "Using occupancy grids for mobile robot perception and navigation". In: *Computer* 22.6 (1989), pp. 46–57.

[21] Steve Fisk. "A short proof of Chvátal's watchman theorem". In: *Journal of Combinatorial Theory, Series B* 24.3 (1978), p. 374.

[22] Rudolf Fleischer et al. "Competitive online approximation of the optimal search ratio". In: *SIAM Journal on Computing* 38.3 (2008), pp. 881–898.

[23] Pierre Fraigniaud et al. "Collective tree exploration". In: *Networks* 48.3 (2006), pp. 166–177.

[24] Subir Kumar Ghosh et al. "Online algorithms with discrete visibility-exploring unknown polygonal environments". In: *IEEE robotics & automation magazine* 15.2 (2008), pp. 67–76.

[25] Yuya Higashikawa et al. "Online graph exploration algorithms for cycles and trees by multiple searchers". In: *Journal of Combinatorial Optimization* 28.2 (2014), pp. 480–495.

[26] Frank Hoffmann et al. "The polygon exploration problem". In: *SIAM Journal on Computing* 31.2 (2001), pp. 577–600.

[27] *hokuyo node*. `http://wiki.ros.org/hokuyo_node`. Accessed: 2017-07-24.

[28] Dirk Holz et al. "A Comparative Evaluation of Exploration Strategies and Heuristics to Improve Them." In: *ECMR*. 2011, pp. 25–30.

[29] Armin Hornung et al. "OctoMap: An efficient probabilistic 3D mapping framework based on octrees". In: *Autonomous Robots* 34.3 (2013), pp. 189–206.

[30] *Joy ROS Package*. `http://wiki.ros.org/joy`. Accessed: 2017-07-24.

[31] Miguel Juliá, Arturo Gil, and Oscar Reinoso. "A comparison of path planning strategies for autonomous exploration and mapping of unknown environments". In: *Autonomous Robots* 33.4 (2012), pp. 427–444.

[32] Brian J Julian, Sertac Karaman, and Daniela Rus. "On mutual information-based control of range sensing robots for mapping applications". In: *The International Journal of Robotics Research* (2014), p. 0278364914526288.

[33] Nathan Koenig and Andrew Howard. "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.* Vol. 3. IEEE. 2004, pp. 2149–2154.

[34] Mathieu Labbe and Francois Michaud. "Appearance-based loop closure detection for online large-scale and long-term operation". In: *IEEE Transactions on Robotics* 29.3 (2013), pp. 734–745.

[35] Mathieu Labbé and François Michaud. "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, pp. 2661–2666.

[36] Joseph SB Mitchell. "Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems". In: *SIAM Journal on Computing* 28.4 (1999), pp. 1298–1309.

[37] Stewart J Moorehead, Reid Simmons, and William L Whittaker. "Autonomous exploration using multiple sources of information". In: *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on.* Vol. 3. IEEE. 2001, pp. 3098–3103.

[38] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Chapman & Hall/CRC, 2010.

[39] *Multimaster fkie ROS Package*. `http://wiki.ros.org/multimaster_fkie`. Accessed: 2017-07-24.

[40] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. "ORB-SLAM: a versatile and accurate monocular SLAM system". In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163.

[41] *Occupancy grid utils*. `https://github.com/clearpathrobotics/occupancy_grid_utils`. Accessed: 2017-07-24.

[42] Joseph O'Rourke. *Art gallery theorems and algorithms*. Oxford University Press Oxford, 1987.

[43] Christian Ortolf and Christian Schindelhauer. "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, pp. 27–36.

[44] Christos H Papadimitriou and Mihalis Yannakakis. "Shortest paths without a map". In: *Theoretical Computer Science* 84.1 (1991), pp. 127–150.

[45] M. Quigley et al. "ROS: an open-source Robot Operating System". In: *ICRA Workshop on Open Source Software*. 2009.

[46] Nagewara SV Rao et al. *Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms*. Tech. rep. Citeseer, 1993.

[47] *RosAria ROS Package*. `http://wiki.ros.org/ROSARIA`. Accessed: 2017-07-24.

[48] Dietmar Schuchardt and Hans-Dietrich Hecker. "Two NP-Hard Art-Gallery Problems for Ortho-Polygons". In: *Mathematical Logic Quarterly* 41.2 (1995), pp. 261–267.

[49] Mac Schwager et al. "A multi-robot control policy for information gathering in the presence of unknown hazards". In: *Proceedings of International Symposium on Robotics Research, Aug*. 2011.

[50] C Stachniss and G Grisetti. *GMapping project at OpenSLAM. org*. 2007.

[51] Sebastian Thrun et al. "Robust Monte Carlo localization for mobile robots". In: *Artificial intelligence* 128.1-2 (2001), pp. 99–141.

[52] *Ubiquiti Networks Bullet*. `https://www.ubnt.com/airmax/bulletm/`. Accessed: 2017-07-24.

[53] Pengpeng Wang, Ramesh Krishnamurti, and Kamal Gupta. "Generalized watchman route problem with discrete view cost". In: *International Journal of Computational Geometry & Applications* 20.02 (2010), pp. 119–146.

[54] Peter Whaite and Frank P Ferrie. "Autonomous exploration: Driven by uncertainty". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.3 (1997), pp. 193–205.

[55] Brian Yamauchi. "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, pp. 146–151.