An Adaptable, Probabilistic Method for Robotic Exploration

Daniel Bigelow and Monica Nicolescu

Abstract— **This paper addresses the problem of robot exploration in the context of answering the question, "Why do we want to explore?" We show that different approaches to improving the performance of the exploration algorithm become necessary depending upon the goals motivating the exploration. We also introduce a probabilistic method for exploration with a single robot that has the ability to easily modify the behavior of the exploring robot to better fit the exploration goals. This method also has the benefit of guaranteeing complete exploration of the environment, and of automatically generating a fully connected and accessible roadmap.**

I. INTRODUCTION

The exploration of unknown environments is an extremely important problem in robotics. This task requires one or more robots to cover an area containing unknown obstacles, while simultaneously learning a model of the environment. Robotic exploration is useful for exploring areas in which humans are unable to move freely, such as outer space, deep ocean, archaeological sites, and disaster areas.

Exploration can be defined as the task of guiding an agent in such a way that it eventually creates a map of every reachable area in the environment. The question then becomes, "How can we make our robot better at exploration?" At its most basic level, exploration involves two steps which are repeated continuously until every reachable part of the environment has been mapped. First, the robot must gather information about the area visible to its sensors. Second, based on the gathered information, the robot must move to a new location in the environment. One of the main challenges in exploration lies in the selection of observation positions to which the robot moves as it explores. Works such as [1], [6] show that different mechanisms for selecting interest points can have dramatic effects on the efficiency of exploration algorithms. One type of exploration algorithm that has been shown to work well overall is frontier-based exploration [12]. This technique keeps track of the boundary between explored and unexplored space, and always selects observation positions along that boundary. This ensures that each observation position will provide some new information to the robot.

We now have an answer to our first question: we can make our robot better at exploration by improving the manner in which it chooses observation positions. However, the reasons motivating exploration greatly affect the choice of exploration algorithms. The question "Why do we explore?" can then have two very different answers. First, we explore in order to find something in the environment. Second, we explore in order to be able to easily move from place to place within the environment. Past research in the area of robotic exploration has focused almost exclusively on minimizing the distance traveled by the exploring robot[12], [6], and thus minimizing the time taken to fully explore the environment. This is ideal, if our motivation for exploring lies in finding something in the environment as quickly as possible. However, if our reason for exploring focuses more on navigating through the environment after it has been explored, there are several ways in which the exploration algorithm can be changed to generate a more useful map.

This paper presents a probability-based approach to frontier-based exploration. Our approach has its foundations in motion planning, and makes use of some of the techniques discussed in [7]. It also draws from previous research done in Sensor-Based Random Trees [9], [5]. This approach generates a roadmap of the environment as the robot explores, which can be used for path-planning once the environment has been explored. Our approach is extremely flexible, making it easy to change the robot's behavior based on the application for which the final map is being used.

The remainder of this paper is organized as follows. Section 2 gives an overview of related work that has been done in the area of robotic exploration. Section 3 discusses our approach to exploration, and the problems that our approach addresses. Section 4 outlines the experiments that were performed to evaluate the effectiveness of our approach, and discusses our results. Section 5 presents conclusions and outlines some potential improvements to our exploration strategy.

II. RELATED WORK

There is a large body of existing work that addresses the problem of robotic exploration. [11] introduces the idea of probabilistic occupancy grid maps, rather than the more common binary occupancy grid. This allows for a more accurate representation of the environment, because it can represent cells in the grid that are only partially obstructed by obstacles. This work also compares different strategies for interest point selection, including closest location, maximum information gain, local maximum information gain, and a weighted combination of closest location and maximum information gain. The results of this comparison show that the best strategies for rapid exploration use a weighted combination of distance traveled by the robot and the expected value of the information gained at each point when selecting observation points. In the same vein, [2] proposes a method

D. Bigelow is with the Department of Computer Science and Engineering, University of Nevada, Reno, NV 89557, USA dbigelow@cse.unr.edu

M. Nicolescu is with the Faculty of Computer Science and Engineering, University of Nevada, Reno, NV 89557, USA monica@cse.unr.edu

for determining the next best interest point by combining traveling cost and expected information gain in a multi-objective function. [1] compares four different strategies for interest point selection. These include random selection, selection based on maximum expected information gain, selection based on a function of distance to travel and expected information gain, and another function of distance and information which uses a more complex model for information gain. In this paper, it is experimentally shown that it is necessary to take into account both distance to travel and expected information gain in order to efficiently explore an environment.

There is also a fairly large body of work focusing entirely on frontier-based strategies for exploration. The seminal work in this direction is [12] proposes an algorithm that operates based only on maximizing the expected information gain, and shows that frontier-based exploration works well for systems with noisy sensors. However, although the frontier-based approach provides a simple method of evaluating the expected information gain of any observation point, it is still important to consider the distance that the robot must travel. [6] evaluates the efficiency of three selection mechanisms within the frontier-based framework: first, choosing the frontier region closest to the robot's current position, second, choosing the closest frontier while constantly checking to see if that frontier disappears due to the motion of the robot (repetitive rechecking), and third, segmenting the map so that individual segments are explored fully before the robot moves on. In this work, it is shown that in open spaces, the repetitive rechecking method works best, since segmentation of open spaces tends to be random and leads to inefficient paths. However, in office-like environments, the presence of numerous rooms makes it relatively easy to segment the map in a meaningful way. In this case, segmentation improves the performance of the algorithm significantly, since it requires the robot to fully explore one room before moving on to another.

[9], [5] introduce the idea of using probabilistic exploration strategies within the framework of the frontier-based algorithm. This has the effect of being simple to implement, and is also proven to be probabilistically complete, so that if the algorithm is allowed to run for a long enough time, it is guaranteed to fully explore the environment. The SRT method discussed in [9], [5] generates a roadmap as it explores the environment, and explores using an algorithm very similar to rapidly-exploring random trees [8].

Finally, [10] pushes for a move away from frontier-based exploration strategies, and explores the idea of using potential fields to guide exploration. This work models exploration as a steady state flow problem over the known workspace, where approximate solutions to the problem provide a scalar field which is guaranteed to bring the robot to the edge of the explored space. In simulated experiments, this approach was shown to out-perform some frontier-based methods in terms of exploration time.

III. APPROACH

A. General Approach

Our method for exploration takes inspiration from [3], [4], where the robot builds a roadmap of the environment as it explores. Our algorithm builds a structure called an Expansive Exploration Tree (EET), which can be viewed as an extension of the Expansive Spaces Tree (EST) proposed in [7]. The EET represents a roadmap of the free configuration space of the robot, where each vertex consists of a collision-free configuration q, as well as a description of the frontier that is local to q. Each vertex is connected to at least one other vertex in the tree via an edge, which represents a collision-free path between the two vertices. The tree is constructed incrementally by selecting a vertex based on some probability density function (PDF), then expanding to a random section of the frontier associated with that vertex. The constraints under which our exploration method can perform are as follows:

1: The workspace in which the robot moves is $\mathbb{R}^2$ or a connected subset of it.
2: The robot is a disk whose configuration q is the position of the disk center. (This allows the configuration space of the robot to be a copy of the workspace with the obstacles grown to allow for robot size.
3: The robot is equipped with sensors which provide LSR(q) or the *Local Safe Region* at q, which is a description of the free space surrounding the robot at q.
4: Each piece of frontier *must* be associated with at least one vertex in the EET.

While these assumptions are necessary for our approach to work, it is reasonable to expect that they are easily enforceable in the real world. Most land-based robots will be moving on a connected subset of $\mathbb{R}^2$, or at least on terrain that can be approximated by $\mathbb{R}^2$. In addition, for the exploration algorithm outlined in this paper, the only hardware needed is a sensor to determine LSR(q), and some form of locomotion, which can easily be contained in a disk-shaped robot. The sensing of LSR(q) can be performed by any number of sensors, such as sonar, infrared laser, or RGBD camera. The only constraint which presents some difficulties is that each piece of frontier must be associated with at least one vertex in the EET. If LSR(q) is being calculated at each time step, and the map is being updated accordingly, the situation in Fig. 1 may arise. In this figure, as the robot traveled from its initial vertex to a point along the frontier, the frontier was pushed back to the edge of the sensor range all along the robot's path. However, this means that there are now frontier ranges that are not within sensor range of any vertex in the EET. This could potentially lead to a situation where frontier exists that is not directly reachable from any vertex in the tree, at which point the algorithm will fail.

Fig. 2 shows our solution to this problem. In this figure, the robot only adds LSR(q) to the map if q corresponds to a vertex in the EET. This means that the robot does not need to employ its sensor unless it is adding a new vertex to the

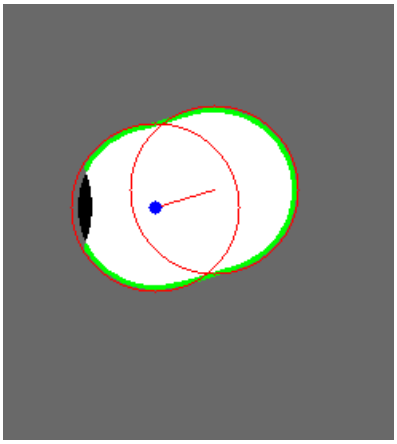

Fig. 1. An example of a case where frontier is unattached to any vertex. The red circles represent the local neighborhood of each vertex, the blue represents the robot's location, and the green represents frontier.

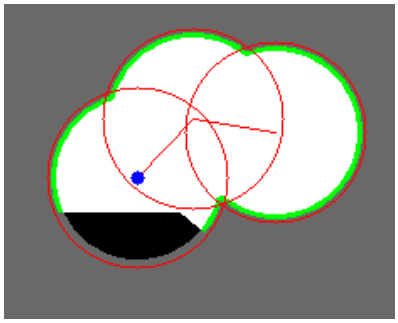

Fig. 2. Our solution to the problem presented in Fig. 1. The red circles represent the local neighborhood of each vertex, the blue represents the robot's location, and the green represents frontier.

EET, which can save on computational overhead. This can be done without fear of collision, since the EET is guaranteed to lie in free configuration space, and the robot is constrained to move either along the EET, or through the LSR associated with a vertex in the EET.

B. Algorithm

Algorithm 1 EET-based exploration

```
perceive LSR
add new vertex to EET
if vertex is not root then
    add edge from current vertex to previous vertex
end if
if frontier exists for any vertex then
    computePDF()
else
    return to start location
    terminate program
end if
choose vertex to expand (based on results of computePDF())
plan path to vertex
travel to target vertex
choose target on local frontier (guaranteed to be in free space)
move to target
```

Algorithm 1 shows the steps of the algorithm used to grow the EET and explore the environment. At each iteration of this algorithm, the Local Safe Region is calculated for the robot's current configuration. This configuration is then added as a vertex to the EET, and the map is updated with the information gained about the LSR and the changes that have occurred in the frontier. At this point, if there is no frontier left in the map, then the environment can be assumed to be fully explored and the robot will return to its initial configuration. However, if there is remaining frontier, then each vertex in the EET will be assigned a weight by the *computePDF()* subroutine, which we discuss more fully below. Once the weights of the vertices have been updated, a vertex will be randomly selected, using the vertex weights as a probability density function. The robot will then travel along the EET until it reaches the selected vertex. Once it arrives at its target, the robot chooses its next observation point on the local frontier. The algorithm then begins again with the addition of a new vertex to the EET, with an edge connecting the new vertex to the vertex most recently visited by the robot.

If our criterion for exploration termination is met, this means that the free space explored by the robot is now completely bounded by impassable terrain, and thus the space has been fully explored. In addition, the EET is guaranteed to possess the qualities needed to constitute a roadmap of the environment. First, it shall be traversable, since it is possible to reach the root of a tree from any vertex. Additionally, since every free configuration in the environment must be contained in the LSR of some vertex of the tree, it will be possible to compute an obstacle-free path from any free configuration q to at least one vertex in the tree. Thus, the tree is also accessible from any free point in the environment. This means that, assuming the EET is well-formed, it will be ideal for path-planning once the environment has been explored, which facilitates our goal of exploring in order to easily move from place to place in an environment.

C. The computePDF() Subroutine

The *computePDF()* subroutine is the part of the algorithm that is responsible for assigning weights to the vertices in the EET. The manner in which this is done drastically affects the behavior of the exploring robot. We explored four different approaches to weighting the EET. One method was to assign a weight of 1 to the closest vertex that had any local frontier remaining, and a weight of 0 to all other vertices. This method replicates the approach to frontier-based exploration mentioned in [6], in which the robot explored by always traveling to the closest frontier. We call this approach Closest Frontier First, or CFF, for the remainder of this paper. We call the next approach Weighting Proportional to Frontier, or

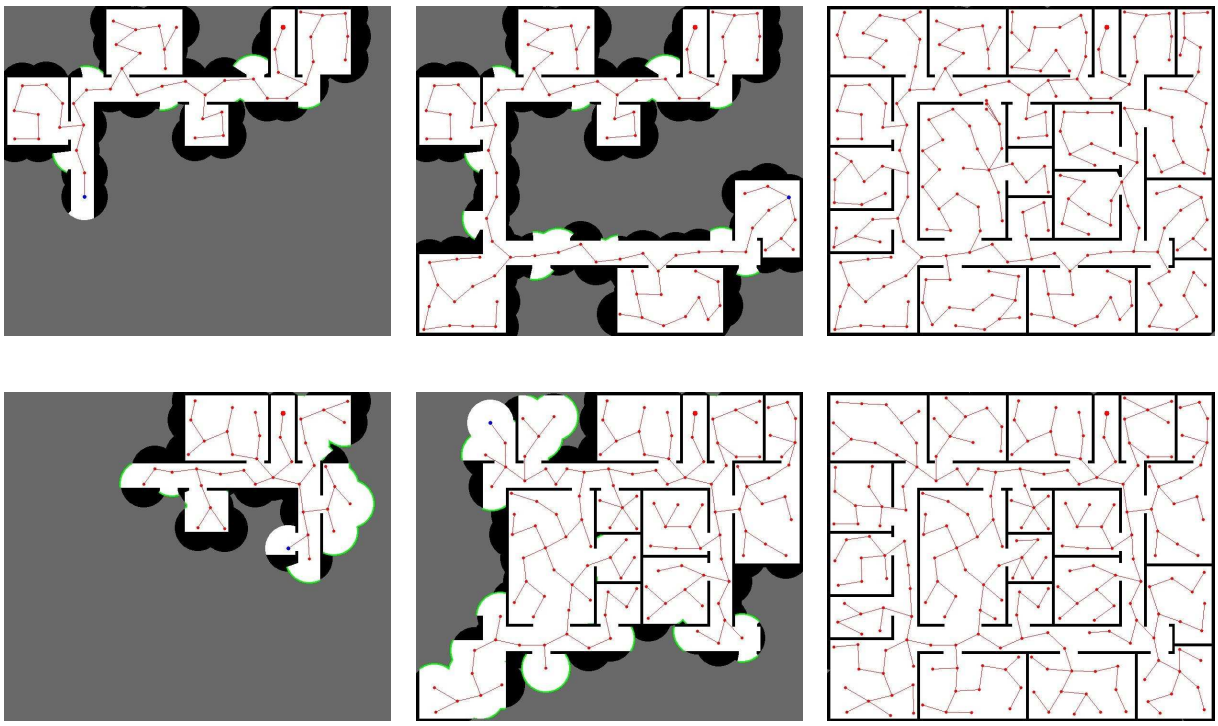

Fig. 3. Incremental snapshots of the exploration process. *Top:* the exploration process when the CFF method of weighting the vertices is used. *Bottom:* the exploration process when the WPF method is used to weight the vertices. Notice the differences in overall tree structure that can be gained by changing *computePDF()*.

WPF. This approach weights each vertex according to the function:

$$w_{WPF}(v) = \frac{LF(v)}{F}$$

where $w_{WPF}(v)$ is the weight assigned to vertex v using the WPF method, $LF(v)$ is the amount of local frontier at vertex v, and F is the total amount of frontier over the entire EET. This approach strives to approximate the potential information gain that would be achieved if the robot were to expand from vertex v.

In an attempt to more accurately represent potential information gain, let us say that $UE(q)$ represents the amount of unexplored space that would become explored if the robot were to place a vertex at configuration q, which we approximate by looking at the area that would be covered by LSR(q), assuming no obstacles were in the LSR. This approach can be called Weighting by Information Gain, or WIG. Now, we can say:

$$ig(v) = \frac{\sum_{q \in LF(v)} UE(q)}{LF(v)} \tag{1}$$

where $ig(v)$ is the expected information gain, computed by taking the average amount of new space that would be explored by placing a vertex at any configuration in $LF(v)$. Then:

$$w_{WIG}(v) = \frac{ig(v)}{\sum_{g \in EET} ig(g)} \tag{2}$$

meaning that the weight assigned to each vertex is calculated by dividing the local expected information gain by the total expected information gain for the entire EET.

Finally, we weight the vertices in the EET based on current distance from the robot and presence of frontier, which we shall refer to as Weighting by Distance, or WD. The idea behind this approach is to encourage the robot to explore areas that are closer to its current location, while not wasting time trying to expand the tree from a vertex which has no local frontier. In this case, we compute:

$$d(v) = \begin{cases} \frac{1}{dist(v,r)} & \text{if } LF(v) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $dist(v, r)$ is the shortest distance between the robot and a vertex v along the EET. We can then compute the weight for each vertex using:

$$w_{WD}(v) = \frac{d(v)}{\sum_{g \in EET} d(g)} \tag{4}$$

As can be seen in Fig. 3, using different methods of weighting the vertices of the EET can result in very different behaviors on the part of the exploring robot. The top row of Fig. 3 shows some steps in the process of exploring by the CFF method, while the bottom row shows the exploration process as performed under the WPF method. As can be seen, the CFF method tends to move through the environment in a depth-first manner, which leads to a fast exploration, but a fairly unbalanced EET. The WPF method, however, tends to expand outwards gradually from the root of the EET, leading to slower exploration times, but a more balanced tree. These differing exploration behaviors shall be discussed further in the following sections.

IV. EXPERIMENTS AND RESULTS

A. Experiments

Testing of the EET algorithm was performed using two simulated environments, in order to obtain repeatable quantitative results, and to be able to run large numbers of experiments in a short amount of time. The environments used were a cluttered environment modeled after an office building, and a more open environment modeled after a conference room with adjoining offices. These are pictured in Fig. 4. In each environment, we ran 100 simulations for each of the *computePDF()* subroutines discussed above. For each simulation, we kept track of the total distance traveled by the robot. This allowed us to evaluate the exploration behaviors in terms of their suitability for exploring quickly, for the case when our reason for exploration is to find something in the environment as quickly as possible. In addition, at the end of each simulation, when the environment was completely explored, we picked 100 pairs of start and end locations at random, computed the shortest path along the EET between each pair, and recorded the average computed path length for each simulation. This allowed us to make some statements about the algorithm's suitability for exploring when our final goal is the ability to move easily through the environment. We also recorded the total number of vertices in each EET and the average branching factor of each EET, in the hopes that we could make some inferences about physical properties of the tree that lead to desirable results in terms of our goals for exploration.

B. Results

Table 1 and Table 2 summarize the results for the office environment and the conference room environment, respectively. They show the average over 100 simulations for the total distance traveled by the robot to fully explore the environment, the average path length computed after each simulation, the average number of vertices in each EET, and the average branching factor of each EET.

Table 1 shows that the CFF method for exploration required the robot to travel a relatively short distance in order to explore the entire environment, while both the WPF and the WIG methods performed very poorly in this area. The WD method did not perform as well as the CFF method, but vastly outperformed both the WPF and WIG approaches. This result can be explained by the behaviors evidenced during exploration. When the CFF method was used, the robot travels with minimum backtracking until it gets as far from the root as possible, then returns to the root, pausing to explore any unexplored regions that are left. In this way, the robot should never travel over the same edge in the tree more than twice. With the WPF and WIG methods, however, the robot will often traverse across the breadth of the entire EET to reach an area with greater potential information gain, leading to a large amount of movement that is wasted on already-explored area. The WD method suffers from the same weakness as the WPF and WIG methods, but is helped by the inclusion of a distance metric, which discourages the robot from traveling long distances across the EET.

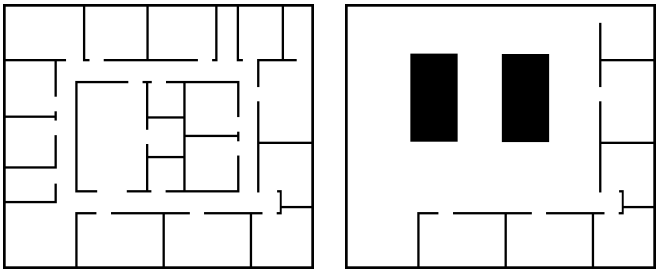

Fig. 4. The two environments used for evaluating the EET method. *Left:* the office-like environment. *Right:* the conference room environment.

TABLE I
RESULTS FOR OFFICE ENVIRONMENT

	CFF	WPF	WIG	WD
Distance Traveled	1488	9263	11443	4779
Avg. Path Length	78.8	56.2	55.8	60.2
Vertices in Tree	233	231	233	235
Avg. Branching Factor	1.25	1.56	1.51	1.49

TABLE II
RESULTS FOR CONFERENCE ROOM ENVIRONMENT

	CFF	WPF	WIG	WD
Distance Traveled	1255	8337	10680	4246
Avg. Path Length	106.2	68.8	63.8	102.2
Vertices in Tree	195	195	198	199
Avg. Branching Factor	1.20	1.55	1.49	1.48

While the CFF method appears to work well for exploring an environment quickly, it is not as good for exploring with the intent of performing path planning once a map is built. Looking at the average path lengths generated when attempting to travel between random pairs of points on the final maps, we see that WPF, WIG, and WD all outperformed the CFF method. If we examine the map construction process shown in Fig. 3, we can see why this might be. While both maps should work equally well for moving from one side of the environment to the other, the very linear nature of the CFF tree means that for more localized start and goal locations, there is a better chance that the robot will be taken far out of its way by the path that is planned along the tree. This is further supported by the average branching factor of the trees generated by CFF, compared to the other three methods. We can see that the branching factor is significantly lower for the CFF method while the total number of vertices in the tree remains fairly constant, which implies that the robot will have less freedom of movement at a local level if this method is used.

Table 2 shows that the performance of the four methods for weighting vertices remains very similar to our results from a cluttered environment in terms of total distance traveled to fully explore, where the CFF method significantly outperforms the WPF and WIG methods, with the WD method falling somewhere in between. However, in this more

open environment, the WD method suffers greatly in terms of suitability for path planning, performing in the same class as the CFF method. This makes sense if we take into consideration the relatively open nature of the environment and the fact that the robot is encouraged to explore closer locations first. In the cluttered environment, exploration of a nearby location would often be curtailed by obstacles, leading the robot to backtrack and expand the EET equally in all directions. This is not the case, however, in more open environments. In this case, the robot is able to explore locally for quite some time before being encouraged to backtrack and grow another branch of the tree. We also see that in an open environment, the average branching factor of the tree has no correlation to the average path length when path-planning on the final map. This makes sense for an open environment given the behavior of the various methods. An analogy to the real world might be that the tree generated by the CFF method is like a single long vine with very few leaves, the tree from the WD method is a single long vine with many leaves, and the WPF and WIG methods generate trees which are multiple short leafy vines all sprouting from a single root. This means that paths found using the CFF and WD trees must travel along the length of the long single vine, while paths found using the other methods can move more quickly from place to place by following a short vine to the root, then another short vine to the goal location.

From our results, we can see that in both cluttered and uncluttered environments, it is better to consider only the distance that the robot must travel when choosing observation points, if our goal is simply to explore the environment as quickly as possible. However, if the goal is to be able to solve motion planning tasks using the generated map once the environment has been explored, it is more important to explore in such a way that the roadmap generated will be well-balanced. In cluttered environments, this can be done without too much loss of speed by taking both distance and potential information gain into consideration. However, in open environments, this is likely to lead to much slower exploration times.

V. CONCLUSIONS AND FUTURE WORK

A. Conclusions

In this paper, we explore the concept of tailoring methods of exploration to fit the use that we have planned for the final map of the environment. We show that methods which work well for exploring an environment quickly will not necessarily work well if our goal is to be able to move efficiently through the environment in the future. We also present an algorithm for exploration that can be easily adapted to fulfill the needs of the user. This algorithm also illustrates that when using probabilistic methods for exploration, one can encourage vastly different behaviors from the exploring robots by simply changing the manner in which the probability density function is calculated. This simple change affects the manner in which the robot chooses observation locations, which in turn can change the suitability of the exploring behavior for the task at hand.

B. Future Work

Possible extensions to our work include expanding the EET algorithm to systems of multiple robots. This would open up the question of which types of PDF over the vertices of the EET work best for both maintaining coordination between individual robots, while also encouraging efficient exploration. It would also raise the question of whether the tradeoff between speed of exploration and quality of roadmap still exists in the multi-robot case.

Additionally, it would be useful to find a way for the robot to automatically detect the type of environment that it is exploring (open vs. cluttered), and adjust its *computePDF()* function accordingly in order to better achieve whatever goals the user may have. For example, if the goal is to explore the environment and be able to path plan using the final map, the robot may want to start out using the WD method, but switch to the WPF method if it detects that it is exploring a more open environment.

REFERENCES

[1] F. Amigoni, *Experimental evaluation of some exploration strategies for mobile robots*, Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, May 2008, pp. 2818 –2823.

[2] F. Amigoni and A. Gallo, *A multi-objective exploration strategy for mobile robots*, Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, April 2005, pp. 3850 – 3855.

[3] Antonio Franchi, Luigi Freda, Giuseppe Oriolo, and Marilena Vendittelli, *A randomized strategy for cooperative robot exploration*, Tech. report, 2006.

[4] Antonio Franchi, Luigi Freda, Giuseppe Oriolo, and Marilena Vendittelli, *A decentralized strategy for cooperative robot exploration*, Proceedings of the 1st international conference on Robot communication and coordination (Piscataway, NJ, USA), RoboComm '07, IEEE Press, 2007, pp. 7:1–7:8.

[5] L. Freda and G. Oriolo, *Frontier-based probabilistic strategies for sensor-based exploration*, Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, April 2005, pp. 3881 – 3887.

[6] Dirk Holz, Nicola Basilico, Francesco Amigoni, and Sven Behnke, *Evaluating the efficiency of frontier-based exploration strategies*, Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK) (2010), 1 –8.

[7] D Hsu, J C Latombe, and R Motwani, *Path planning in expansive configuration spaces*, International Journal of Computational Geometry and Applications **9** (1997), no. April, 495–512.

[8] Steven M. LaValle, James J. Kuffner, and Jr., *Rapidly-exploring random trees: Progress and prospects*, 2000.

[9] G. Oriolo, M. Vendittelli, L. Freda, and G. Troso, *The srt method: randomized strategies for exploration*, Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on, vol. 5, April-1 May 2004, pp. 4688 – 4694 Vol.5.

[10] Robbie Shade and Paul Newman, *Choosing where to go : Complete 3d exploration with stereo*, 2011 IEEE International Conference on Robotics and Automation (2011), 2806–2811.

[11] Cyrill Stachniss and Wolfram Burgard, *Exploring unknown environments with mobile robots using coverage maps*, 2003.

[12] Brian Yamauchi, *A frontier-based approach for autonomous exploration*, In Proceedings of the IEEE International Symposium on Computational Intelligence, Robotics and Automation, 1997, pp. 146–151.