A Multi-robot Strategy for Rapidly Searching a Polygonal Environment

Alejandro Sarmiento, Rafael Murrieta-Cid and Seth A. Hutchinson

Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign
Urbana Illinois, USA
{asarmien, murrieta, seth}@uiuc.edu

Abstract. In this paper we address the problem of finding an object in a polygonal environment as quickly as possible on average, with a team of mobile robots that can sense the environment.
We show that for this problem, a trajectory that minimizes the distance traveled may not minimize the expected value of the time to find the object. We prove the problem to be NP-hard by reduction, therefore, we propose the heuristic of a utility function. We use this utility function to drive a greedy algorithm in a reduced search space that is able to explore several steps ahead without incurring too high a computational cost. We have implemented this algorithm and present simulation results for a multi-robot scheme.

1 Introduction

The problem of determining a good strategy to accomplish a visibility-based task such as environment modeling [4], pursuit-evasion [7] [8], or object finding [6] [15], is a very challenging an interesting research area. Specially when the sensors are not static but rather are carried by mobile robots.

In this paper we address the problem of finding an object in an unknown location somewhere inside a polygonal environment as quickly as possible on average. For this, we use a team of mobile robots that can sense the environment. This is the optimization problem of minimizing the expected value of time required to find the object, where time is a random variable defined by a search path together with the probability density function associated to the object's location. The possible applications have a wide range, from finding a specific piece of art in a museum to search and rescue of injured people inside a building.

We present a discrete formulation, in which we use a visibility-based decomposition of the polygon to convert the problem into a combinatoric one. We define particular locations from where the robot senses the environment (guards). The guards' visibility regions are used to calculate the probability of seeing an object for the first time from a particular location. These are chosen from an appropriate set which is computed automatically. With this, we are able to abstract the problem to one of finding a path in a graph whose nodes represent the sensing locations. Trajectories are then constructed from arcs in a reduced visibility graph.

We show that for this problem, a trajectory that minimizes the distance traveled may not minimize the expected value of the time to find the object. We prove the problem to be NP-hard by reduction, therefore, we propose the heuristic of a utility function, defined as the ratio of a gain over a cost. We use this utility function to drive a greedy algorithm in a reduced search space that is able to explore several steps ahead without incurring too high a computational cost. We have implemented this algorithm and present simulation results for a multi-robot scheme.

2 Problem Definition

In general terms, we define the problem of searching for an object as follows: Given one or more mobile robots with sensing capabilities, a completely known environment and an object somewhere in the world, develop a motion strategy for the robots to find the object in the least amount of time on average.

The environment W is known, and modeled as a polygon that may contain holes (obstacles). The obstacles generate *both motion and visibility constraints.*

Furthermore, we assume that the probability of the object being in any specific point is uniformly distributed throughout the polygon's interior. Therefore, the probability of the object being in any subset $R \subseteq W$ is proportional to the area of R.

We also assume that we start with a set of locations L (also known as *guards*, from the art gallery problem [14]) so that each point in W can be seen from at least one location in L. There are several criteria for determining the goodness of this set. For example, the minimal number of locations (art gallery problem [9]), locations along the shortest path that covers the whole environment (shortest watchman path [14]), and so on. In [13] we propose an algorithm for determining this set. The basic idea is to place guards inside the region bounded by inflection rays of the aspect graph. These regions have the property that a point inside can see both sides of reflex vertices (those with internal angle greater than π).

The visibility region of location L_j, denoted $V(L_j)$, is the set of points in W that have a clear line of sight to L_j (the line segment connecting them does not intersect the exterior of W). The set L is chosen so that the associated visibility regions define a cover of W. This means that their union is to the whole environment, that is, $\bigcup_j V(L_j) = W$. We do not require nor assume the set L to be minimal.

For the sake of simplicity, *we will first present the basic algorithm for the case of a single robot and then we will extend it for the general multi-robot case.*

Our exploration protocol is as follows: the robot always starts at a particular location in L (the starting point) and visits the other locations as time progresses (it follows the shortest paths between them). It only gathers information about the environment (sensing) when it reaches one of these locations – it does not sense while moving. We describe the route followed by the robot as a series of locations L_{i_k} that starts with the robot's initial location and includes every other location at least once. Note that while L_j refers to locations in the environment,

L_{i_k} refers to the *order* in which those locations are visited. That is, the robot always starts at L_{i_0}, and the k-th location it visits is referred to as L_{i_k}.

For any route R, we define the time to find the object T as the time it takes to go through the locations – in order – until the object is first seen.

Our goal is to find the route that minimizes the expected value of the time it takes to find the object

$$E\left[T|R\right] = \sum_j t_j P\left(T = t_j\right) \tag{1}$$

where

$$P\left(T = t_j\right) = \frac{Area\left(V\left(L_{i_j}\right) \setminus \bigcup_{k<j} V\left(L_{i_k}\right)\right)}{Area(W)}. \tag{2}$$

Here, t_j is the time it takes the robot to go from its initial position – through all locations along the route – until it reaches the j-th *visited* location L_{i_j}, and $P\left(T = t_j\right)$ is the probability of seeing the object for the first time from location L_{i_j}. Since the robot only senses at specific locations, we also denote this probability of seeing the object for the first time from location L_{i_j} as $P\left(L_{i_j}\right)$.

Explicitly, the probability of seeing the object for the first time from a given location is proportional to the visibility polygon of that region $V\left(L_{i_j}\right)$ minus the already explored space up to that point $\bigcup_{k<j} V\left(L_{i_k}\right)$.

2.1 Expected Value vs. Worst Case

It is important to note the difference between minimizing the expected value of the time to find an object and minimizing the time it would take in the worst case. To minimize the worst case time, the robot must find the shortest path that completely covers the environment (the Shortest Watchman Tour problem [1]). This usually means that no portions of the environment are given any priority over others and the rate at which new portions of the environment are seen is not important.

On the other hand, to minimize the expected value of the time, the robot must gain probability mass of seeing the object as quickly as possible. For a uniform object PDF, this translates into sensing large portions of the environment as soon as possible, even if this means spending more time later to complete covering the whole environment. For non-uniform PDFs, the robot should visit the most promising areas first. We believe this represents another paradigm for coverage tasks, where it is important to gain as much new information in the shortest time as possible. This could be very useful in applications where the time assigned to the task is limited or not completely known.

The trajectories that satisfy the previous two criteria are not the same. In fact, for a given environment, the route that minimizes the distance traveled may not minimize the expected value of the time to find an object along it. Consider the example in Fig. 1.

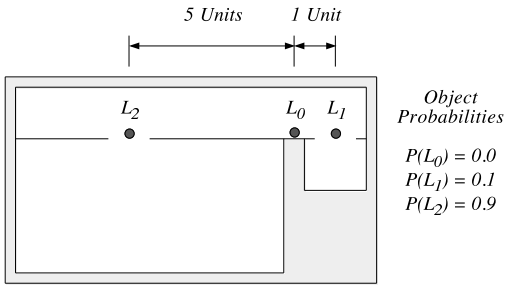

Fig. 1. Example with a simple environment

The robot starts in the corridor at location L_0. The object will always be in one of two rooms, and the probability of it being in either is related to the size of the room.

If the robot goes to the smaller room first and then moves on to the larger room (route 1), it reaches L_1 at time 1 and L_2 at time 7. The expected value of the time it takes to find the object following this route is $E[T|(L_0, L_1, L_2)] = (0.1)(1) + (0.9)(7) = 6.4$. The robot always completes its search after 7 seconds.

On the other hand, if the robot moves to the larger room first and then goes to the smaller room (route 2), it reaches L_2 at time 5 and L_1 at time 11. The expected time in this case is $E[T|(L_0, L_2, L_1)] = (0.9)(5) + (0.1)(11) = 5.6$. In the worst case, it will take the robot 11 seconds to find the object.

A robot following route 1 always finishes searching after 7 seconds, while a robot following route 2 takes 11 seconds. Route 1 minimizes the distance traveled. However, the average time it takes for a robot following route 1 to find the object is 6.4 seconds whereas for route 2 it is only 5.6 seconds. Route 2 minimizes the expected value of the time to find an object.

Thus, *a trajectory that is optimal in the distance traveled does not necessarily minimize the expected value of the time to find an object along it.*

3 Proposed Solution

Since we assume that we are given a set of sensing locations that completely cover the environment, we are interested in finding an order of visiting those locations – the problem becomes a combinatorial search.

In general, the robot will not be able to travel between two locations by following a straight line. In this cases, we use a reduced visibility graph [10] and Dijkstra's Algorithm to follow the shortest path between them.

3.1 Reduction from an NP-hard problem

The *Minimum Weight Hamiltonian Path Problem*, known to be NP-hard [3], can be reduced to the problem of finding the optimal visiting order of sensing locations which minimizes the expected time to find an object.

In order to make a formal reduction, we abstract the concept of environment and visibility regions. We only consider a set of locations that have an associated probability of seeing the object and whose visibility regions do not overlap.

The reduction consists in defining the distance between the sensing locations as the edge weights of the Minimum Weight Hamiltonian Path Problem and setting the probabilities uniformly (same value for all).

Since the probabilities are set uniformly, the route that minimizes the expected time will be exactly the same as the one that minimizes the distance traveled. This happens because the expected value of the time to find an object is determined only by the time it takes to reach locations along the route. Since time is proportional to distance, the route that minimizes time will also minimize the distance.

Given that the solutions to both problems are the same ordering of locations, finding a polynomial algorithm to solve these instances of the defined problem would also solve the Minimum Weight Hamiltonian Path Problem in polynomial time, thereby proving that the proposed problem is NP-hard.

3.2 Utility Heuristic

Since trying to find an optimal solution is a futile effort, we have decided to implement an iterative, greedy strategy, one that tries to achieve a good result by exploring just a few steps ahead.

We propose a greedy algorithm, called *utility greedy*, that tries to maximize a utility function. This function measures how convenient it is to visit a determined location from another, and is defined as follows:

$$U\left(L_j, L_k\right) = \frac{P\left(L_k\right)}{Time\left(L_j, L_k\right)}. \tag{3}$$

This means that if the robot is currently in L_j, the utility of going to location L_k is proportional to the probability of finding the object there and inversely proportional to the time it must invest in traveling.

A robot using this function to determine its next destination will tend to prefer locations that are close and/or locations where the probability of seeing the object is high. Intuitively, it is convenient to follow such a strategy, but its relationship with the expected value minimization will be more evident after the following analysis.

Consider a definition of expectation for a non-negative random variable, such as time, from [11]

$$E[T|R] = \int_0^\infty P(T > t)dt = \int_0^\infty \left(1 - P(T \leq t)\right) dt = \int_0^\infty \left(1 - F_T\right) dt \tag{4}$$

in which F_T is a *cumulative distribution function*.

In our problem, every valid trajectory R defines a particular cumulative distribution function of finding the object, F_T. Since we are dealing with a discrete problem, the distributions are only piecewise continuous with the discontinuities

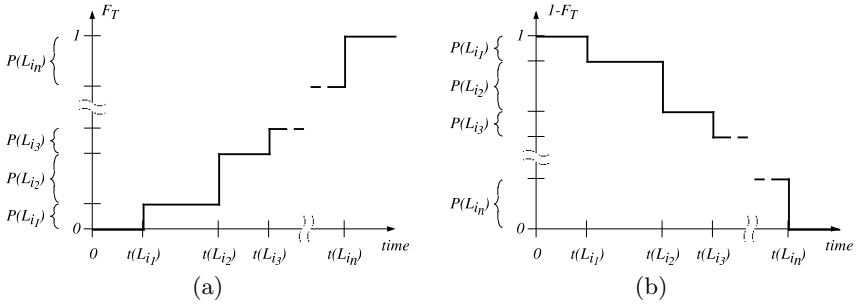

Fig. 2. Defined cumulative distribution functions. (a) F_T (b) $1 - F_T$

being the times at which the robot reaches the distinct locations along the route, as shown in Fig. 2a.

By (4), we know that the expected value of a random variable with distribution F_T is the area under the curve $1 - F_T$, shown in Fig. 2b. This area is the value we want to minimize.

One method for making this area small is to have the time intervals as small as possible and the probability changes (down step) as large as possible. This is the notion that our utility function in (3) captures; its value is larger when the probability of seeing the object from a particular location is high (large down step) and/or when the location is near (small time interval).

3.3 Efficient Utility Greedy Algorithm

The utility function in (3) is sufficient to define a 1-step greedy algorithm. At each step, simply evaluate the utility function for all available locations and choose the one with the highest value. This algorithm has a running time of $O\left(n^2\right)$. However, it might be convenient to explore several steps ahead instead of just one to try to "escape local minima" and improve the quality of the solution found. The downside of this idea is that it usually increases the complexity of the algorithm by a factor of $O(n)$ for each look ahead step.

To reduce this effect we propose a second heuristic that reduces the branching factor. The heuristic is that the children of each location can only be those other locations that are not strictly dominated according to the two variables in the utility function $P\left(L_k\right)$ and $Time\left(L_j, L_k\right)$. As seen from the j-th location L_j, a location L_k strictly dominates another L_l if both of the following conditions are true

$$P\left(L_k\right) > P\left(L_l\right),$$
$$Time\left(L_j, L_k\right) < Time\left(L_j, L_l\right).$$

It is straightforward that dominating locations will lie on the convex hull of the remaining set of locations when plotted on the probability vs. distance plane.

The endpoints of this partial convex hull are not considered as candidates since they are not defined locations.

The final algorithm for a single robot is as follows:

(1) For the last location along the current solution (initially just the robot starting location) explore the possible routes (create a tree breadth-first) until the number of nodes is of order $O(n)$.
(2) For each node that needs to be expanded, compute the set of locations that are not strictly dominated by others and only choose those as children. This can be done with a convex hull algorithm with complexity $O(n \log n)$.
(3) When the number of nodes in the exploration tree has reached order $O(n)$, choose the best leaf according to the heuristic in (3), discard the current tree and start over with the best node as root.

The complexity of the algorithm is proportional to exploring a tree of order $O(n)$, choosing the best children for each node in the tree with a convex hull algorithm in $O(n \log n)$ and repeating $\frac{n}{\log n}$ times to generate a complete route. This is

$$O\left(n \cdot n \log n \cdot \frac{n}{\log n}\right) = O\left(n^3\right).$$

Of course, this result depends on the number of dominating locations being significantly smaller than n on average, which may be difficult to determine for a specific problem. We know, for example, that the expected number of points on the convex hull of a set sampled uniformly from a convex polygon is of order $O(k \log n)$ for a k-sided polygon [2]. In the worst case, when the branching factor is not reduced at all, our algorithm only explores one step at a time and has a running time of

$$O(n \cdot n \log n \cdot n) = O\left(n^3 \log n\right). \tag{5}$$

3.4 The General Multi-Robot Case

For the case of a single robot, each sensing location determines a state for the environment search. Each node in the search graph corresponds exactly to one location. For the case of multiple robots, the state has to be augmented to include the status of every robot in the team. Each robot can be performing one of two possible actions: traveling or sensing. Therefore, a node in the search graph now corresponds to a n-tuple of robot actions. We assume that two or more robots will never arrive to sensing locations exactly at the same time, which is true for sensing locations that are in general position. For example, the n-tuple (T_4, S_{17}, T_8) represents a state in which the first robot (first location in the tuple) is traveling towards location L_4, the second robot is sensing at location L_{17} and the third robot is moving to location L_8.

This approach allows us to discretize time into uneven intervals bounded by critical events. These events correspond to the times robots reach given locations and sense the environment. This formulation defines a new state space which is searched with the same utility function algorithm proposed for a single robot.

Under our scheme, the possible next states only consider the locations already visited by all the robots, and once a robot committs to going to a certain location, it will not change regardless of what the other robots are doing. This yields a scheme which guarantees that one more location will be visited at each exploration step. This means that the computational complexity for a team of robots is exactly the same as for the single robot case stablished in (5). However, this scheme will not explore all possible path permutations (which would be exponential), only a reduced state space.

3.5 Probability Computation for Polygonal Environments

We assume a uniform probability density function of the object's location over the environment, consequently, the probability of seeing the object from any given location is proportional to the area of the visibility region from that location (point visibility polygon [2]). This visibility polygon can be computed in linear time to the number of vertices in simple polygons and in $O(n \log n)$ time for general polygons [2]. If the results are cached, this has to be done only once for each location.

The probability of seeing the object for the first time at location L_{i_j}, denoted $P(T = t_j)$, is proportional to the area visible from L_{i_j} minus the area already seen from locations L_{i_k} $\forall k < j$, as stated in (2). This involves polygon clipping operations of union and difference. We know that any clipping algorithm supporting two arbitrary polygons must have a complexity of at least $O(n\ m)$ where n and m are the number of vertices in each polygon [5].

The cost of performing these clipping operations must be added to the complexity in (5) to describe the total complexity of the algorithm when applied to general polygons.

4 Simulation Results

For our simulations, we implemented routines for computing visibility polygons, the reduced visibility graph and shortest paths (Dijkstra's Algorithm). For calculating the union of visibility regions, we used the *gpc* library developed by Alan Murta based on an approach proposed in [16].

Fig. 3 shows the paths generated by our proposed approach for two different environments and a team of two robots. Parts (a) and (b) show the environments (black obstacles), the set of sensing locations (crosses) and the initial position (black disc). Parts (c) and (d) show the final paths and different regions sensed by the robots. The light grey path corresponds to robot 1, and the dark grey to robot 2. The dark grey area was only sensed by robot 1, the light grey only by robot 2 and the white area was seen by both.

As can be seen, the robots split the effort of sensing the environment in approximately equal amounts, even when an intuitive partition is not evident, as in the case of the environment shown in the right column of Fig. 3.

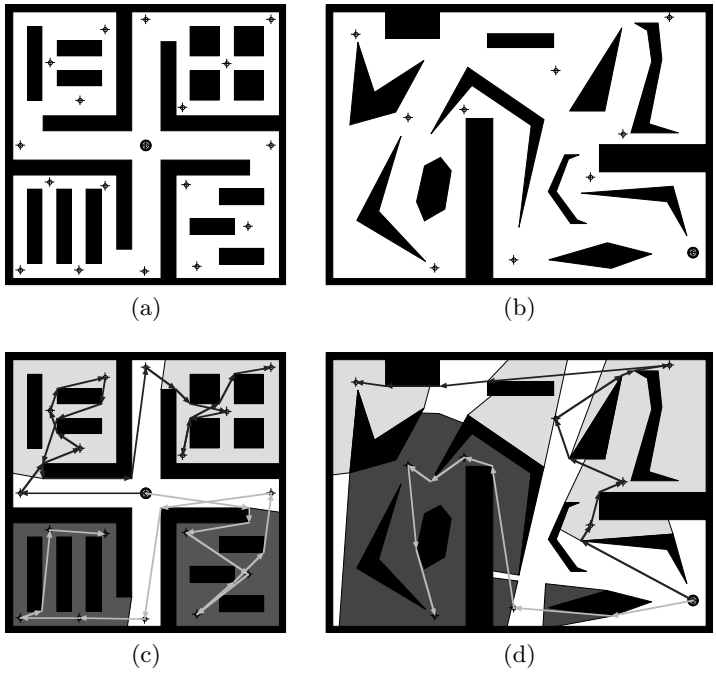

Fig. 3. Simulation results for two environments and a team of two robots

For the case of a single robot, the results obtained by our algorithm are close to the optimal case but with a major improve in computation time [12]. In some instances, we have observed over a thousand-fold improvement. For a team of multiple robots, the expected value of the time is further reduced, as expected, *but with the exact same order of computational complexity.*

5 Conclusions and Future Work

In this paper we proposed an efficient approach to solve the problem of finding an object in a polygonal environment as quickly as possible on average. We proposed the heuristic of a utility function to drive a greedy algorithm in a reduced search space that is able to explore several steps ahead without incurring too high a computational cost. We implemented this algorithm and presented simulation results for a multi-robot scheme.

In [13] we address the problem of continuous sensing for expected value search with a single robot. For this, we use the calculus of variations to compute locally optimal sub-paths and concatenate them to construct complete trajectories. As future work, we plan on extending the continuous sensing case to multiple robots.

Currently, we are also working on the case where the environment is three dimensional and the robot is a seven degree of freedom mobile manipulator with an "eye-in-hand" sensor, like the one shown in Fig. 4.

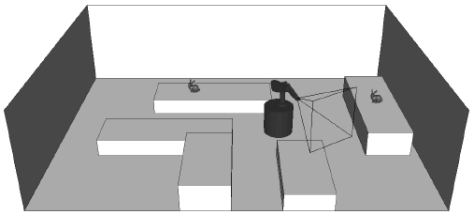

Fig. 4. A 3-D workspace with a mobile manipulator

References

1. Chin, W.P. and S. Ntafos, "Optimum Watchman Routes," *Information Processing Letters,* Vol. 28, pp. 39–44, 1988.
2. Goodman, J.E. and J. O'Rourke, *Handbook of Discrete and Computational Geometry,* CRC Press, 1997.
3. Garey, M.R. and D.S. Johnson, *Computers and Intractability,* W. H. Freeman and Company, 1979.
4. González-Baños, H.H. and J.C. Latombe, "Navigation Strategies for Exploring Indoor Environments," *Int. Journal of Robotics Research,* 21(10/11), pp. 829–848, Oct–Nov 2002.
5. Greiner, G. and K. Hormann, "Efficient Clipping of Arbitrary Polygons," *ACM Transactions on Graphics,* Vol. 17, number 2, pp. 71–83, 1998.
6. LaValle, S.M. *et al*, "Finding an Unpredictable Target in a Workspace with Obstacles," *in Proc. IEEE Int. Conf. on Robotics and Automation 1997.*
7. Murrieta-Cid, R., A. Sarmiento and S.A. Hutchinson, "On the Existence of a Strategy to Maintain a Moving Target within the Sensing Range of an Observer Reacting with Delay," *in Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems 2003.*
8. Murrieta-Cid, R., A. Sarmiento, S. Bhattacharya and S.A. Hutchinson, "Mantaining Visibility of a Moving Target at a Fixed Distance: The Case of Observer Bounded Speed," *in Proc. IEEE Int. Conf. on Robotics and Automation 2004.*
9. O'Rourke, J., *Art Gallery Theorems and Algorithms,* Oxford University Press, 1987.
10. Rohnert, H., "Shortest Paths in the Plane with Convex Polygonal Obstacles," *Information Processing Letters,* 23:71–76, 1986.
11. Ross, S.M., *Introduction to Probability and Statistics for Engineers and Scientists,* Wiley, 1987.
12. Sarmiento A., R. Murrieta and S.A. Hutchinson, "A Strategy for Searching an Object with a Mobile Robot," *in Proc. Int. Conf. on Advanced Robotics 2003.*
13. Sarmiento A., R. Murrieta and S.A. Hutchinson, "Planning Expected-time Optimal Paths for Searching Known Environments," *submitted to IEEE/RSJ Int. Conf. on Intelligent Robots and Systems 2004.*
14. Shermer, T.C., "Recent Results in Art Galleries," *in Proc. of the IEEE,* Vol. 80, issue 9, September 1992.
15. Tovar, B., S.M. LaValle and R. Murrieta-Cid, "Optimal Navigation and Object Finding without Geometric Maps or Localization," *in Proc. IEEE Int. Conf. on Robotics and Automation 2003.*
16. Vatti, B.R., "A Generic Solution to Polygon Clipping," *Communications of the ACM,* 35(7), pp. 56–63, July 1992.