Randomized Algorithms for Minimum Distance Localization

Malvika Rao, Gregory Dudek, and Sue Whitesides

McGill University, Montréal, Canada H3A 2A7 {rao,dudek,sue}@cs.mcgill.ca

Summary. *We address the problem of minimum distance localization in environments that may contain self-similarities. A mobile robot is placed at an unknown location inside a 2D self-similar polygonal environment P. The robot has a map of P and can compute visibility data through sensing. However, the self-similarities in the environment mean that the same visibility data may correspond to several different locations. The goal, therefore, is to determine the robot's true initial location while minimizing the distance traveled by the robot. We present two randomized approximation algorithms that solve minimum distance localization. The performance of our algorithms is evaluated empirically* [1].

1 Introduction

This paper addresses the problem of minimum distance localization. That is, the problem of estimating a robot's position in an environment without a prior position estimate and of selecting a minimum length path along which to acquire sensor data to accomplish this localization as quickly as possible. Global localization refers to estimating the robot's position where the space of possibilities is the entire environment [2]. In general, global localization must contend with the fact that the observations used to estimate position can be ambiguous (even with a perfect map and a perfect sensor, this ambiguity can be serious [6]). Thus, global localization intrinsically entails combining measurements from multiple vantage points along some path. The problem of selecting an optimal (minimum length) trajectory for estimating position is NP-hard [3]. Simple heuristic strategies have been used in practice but these can be shown to have strikingly poor performance. This paper presents algorithms for determining a localizing path that are efficient to compute and yet also efficient in their performance.

Existing deterministic solutions for calculating even nearly optimal localizing paths are computationally costly in complex environments. Further,

[1] Research supported in part by NSERC and FCAR.

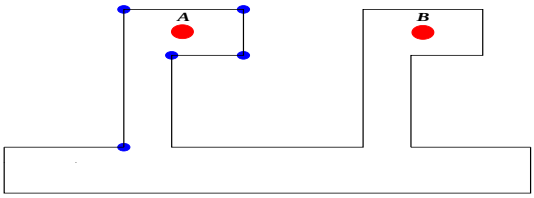

Fig. 1. Visible vertices from location A. The visibility data at locations A and B are the same.

previous work in minimum distance localization [3] may require the robot to make observations that are arbitrarily difficult to achieve in practice. That is, the robot is directed to visit a series of visibility cells - places where specific combinations of landmarks in the environment can be seen - even though these cells may be arbitrarily small. For any real robot with any odometry error whatsoever, maneuvering into such cells may not be feasible.

The algorithms we propose are based on visiting a series of randomly selected sample points in the environment from which distinguishing landmarks may be observed. In the context of our environment model, and with full generality, we consider landmarks to be combinations of vertices and edges that define the perimeter of the environment. Out of a set of randomly selected points in the environment, the robot visits the location that promises the most information gain, where it collects sensor data in order to disambiguate its initial position. We argue that by virtue of random sampling, our strategy would direct the robot to visit a particular visibility cell with a probability directly proportional to the area of that cell. Hence vanishingly small visibility cells have vanishingly small probabilities of being visited.

The performance of our randomized localization algorithms is validated and explored via experiments on a range of simulated environments. Our results show that path length improves rapidly as the number of random sample points is increased, and eventually settles at the near-optimum value. Moreover, for a given environment, computing time appears to be linear with respect to the number of sample points.

Although our work deals with the idealized case of a perfect polygonal map and an ideal range sensor, our algorithms are flexible and can generalize easily to more realistic and challenging environments. These results are significant for many problems in which pose estimation must take place in the context of ambiguity, including certain cases of SLAM (simultaneous localization and mapping) where pose errors may become large.

The rest of this paper is organized as follows: The next section discusses related work. Section 3 provides a formal description of the problem and states the assumptions underlying our approach. Our randomized localization algorithms are presented in Section 4. Experimental results are described in Section 5. Finally, we close with a discussion of our work and directions for future research.

2 Related Work

Dudek, Romanik, and Whitesides [3] present an approximation algorithm in which the localization problem is treated in two phases: hypothesis generation and hypothesis elimination. For the hypothesis generation phase, the solution proposed by Guibas, Motwani, and Raghavan [6] is used. Guibas *et al.* show how to preprocess a map polygon P, by computing its visibility cell decomposition, so that given the robot's visibility polygon V, the set of points in P whose visibility polygon is congruent to V and oriented similarly can be returned. This technique preprocesses P in $O(n^5 \log n)$ time and $O(n^5)$ space, where n is the number of vertices of P. For hypothesis elimination, an overlay arrangement is computed yielding $O(n^6)$ cells. The robot then greedily travels to the nearest distinguishing visibility cell from its initial location in order to rule out hypotheses. By measuring distance with respect to the robot's initial location, rather than its current location, the paper is able to show that the robot travels a path of at most $(k-1)d$, where k is the number of initial hypothetical locations generated and d is the length of an optimal verification tour. The authors prove that this competitive ratio is the best possible. However, with space and time complexity of $O(n^6)$, their algorithm is impractical to use. The authors also demonstrate that simple heuristic localization strategies can sometimes exhibit strikingly poor performance.

Schuierer [8] proposes a technique that uses geometric overlay trees to speed up the implementation of the greedy localization strategy put forth by [3]. While his approach reduces the time complexity to $O(kn^2)$ and space complexity to $O(kn)$, no implementation results are given and it is unclear how to extend the technique to address more practical issues. Kleinberg [7] approaches robot localization by modeling the environment as a bounded geometric tree. Brown and Donald [1] describe algorithms for robot localization which allow for uncertainty in the data returned by the range sensor. Fox, Burgard, and Thrun [4] use Markov localization to determine the position of a mobile robot from sensor data. In that work, global localization is achieved, but the length of the localizing trajectory relative to the optimum is not considered.

Gonzalez-Banos and Latombe [5] present randomized algorithms that use a $2D$ map to estimate the locations where sensing will be most effective through random sampling of the workspace. Their work does not address the problem of robot localization.

3 Problem Specification

In this section we formally define the localization problem and state our assumptions about the robot and its environment. We are given a random environment modeled by an n-vertex simple polygon P without holes positioned somewhere in the $2D$ plane. A mobile robot is placed at an unknown initial

location within P, of which it has a map. First, the robot must determine if its initial location is unique by sensing its surroundings and matching the resulting visibility data W to the map of the environment. Given P and W, the robot must generate the set H of all hypothetical locations $p_i \in P$ such that the visibility at p_i is congruent under translation to W. Next, the robot must determine its true initial location by sensing and traveling in order to eliminate all hypothetical locations but one from H, while minimizing the distance traveled.

The robot's sensor behaves as a "perfect" sensor in that it can detect distances to those points on the boundary of the environment for which the robot has an unobstructed line of sight. The visibility data W sensed by the robot is composed of the counter-clockwise ordering of vertices and edges seen by the robot (see Figure 1). Geometric relationships amongst the data such as vertex angles, distances, adjacencies, and the robot's relative position with respect to the data sensed are available. W can also be regarded as a visibility polygon.

The robot is assumed to be a point robot moving in this static $2D$, obstacle-free environment. The robot is able to make error-free motions between arbitrary locations in the environment. The movement of the robot is restricted to the inside and along the boundary of the environment. As well, it is able to determine its orientation. Otherwise it would be impossible for the robot to uniquely determine its exact location in an environment with non-trivial symmetry such as a square [2].

4 Approach

Our localization algorithms (like that of [3]) comprise two phases: hypothesis generation and hypothesis elimination. The hypothesis generation phase computes the set of hypothetical locations that match the observations sensed by the robot at its initial location. The hypothesis elimination phase rules out incorrect hypotheses thereby determining the true initial location of the robot. However, unlike the hypothesis generation phase of [3], ours does not involve any preprocessing of the map polygon P, nor is the visibility cell decomposition of P computed. Instead we generate hypotheses using an online method. Our hypothesis elimination phase also differs from that of [3]. In contrast to the deterministic evaluation visibility cells as a potential probe locations, carried out in [3], we choose potential probe locations by randomly sampling points in certain regions of P. We then check if the sampled location provides any new information. This avoids the computational complexity of calculating the visibility cell decomposition together with the overlay arrangement as performed in [3]. Moreover, rather than pursue a greedy choice as in [3], our decision strategy directs the robot to make a weighted choice with respect to its initial location. The utility or information gain of each potential probe

[2]Note: the robot could still solve the problem up to rotational symmetry.

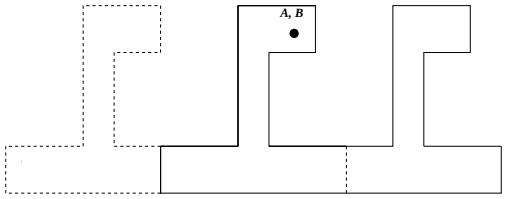

Fig. 2. Overlay arrangement for the situation shown in Figure 1.

destination is weighted by its distance from the robot's initial location. Before we continue with a description of the localization algorithms, we describe some key definitions and procedures.

4.1 Hypothesis Generation

Given an environment P and visibility data W, we want to determine the set H of all hypothetical locations $p_1, p_2, ..., p_k \subset P$ such that the visibility data computed at p_i, $Vis(p_i)$, matches W for $i = 1, ..., k$ (see Figure 1). This is done by tracing the pattern of vertices and edges listed in W in the set of vertices and edges of P. We search for as many sets of vertices and edges in P that match W as possible. Note that k refers to the number of *initial* hypothetical locations generated.

4.2 Overlay Intersection

The hypothesis generation phase generates a set $H = p_1, p_2, ..., p_k \subset P$ of hypothetical locations in P at which the robot might be located initially. Without loss of generality, we select an arbitrary hypothetical location p_i from H to serve as a reference point or origin. Next, for each hypothetical location $p_j, 1 \leq j \leq k$, a translation vector $t_j = p_i - p_j$ is defined that translates location p_j to p_i. As a result, we compute a set of copies $P_1, P_2, ..., P_k$ of the environment polygon P, corresponding to the set of hypothetical locations H, such that P_j is congruent to P translated by the vector t_j. Copy P_i is translated by the zero vector. The point in each translated polygon P_j corresponding to the hypothetical location p_j is now located at the origin p_i. We can now define the overlay arrangement as follows:

Definition 1 (Overlay Arrangement [3]).
The overlay arrangement *for the environment polygon P corresponding to the set of hypothetical locations H is the structure obtained by taking the union of the edges of each translated polygon $P_j, 1 \leq j \leq k$.*

Figure 2 illustrates an overlay arrangement for the situation shown in Figure 1. We consider only the *connected component of the intersection region of the overlay arrangement that contains the origin* since it is the area known to

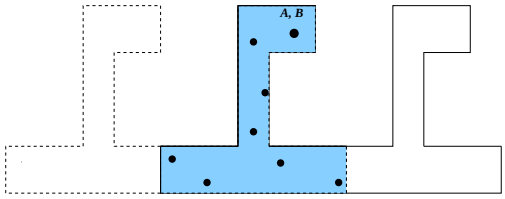

Fig. 3. Shaded area represents the overlay intersection region OI. Random points are chosen within this region.

exist in all hypotheses (see Figure 3). We will refer to this connected overlay intersection component containing the origin as OI. In this respect, we differ from [3] in recognizing that clearly, we do not need to examine visibility cells lying outside this overlay intersection region, common to all hypothetical locations, for new information. We compute OI by performing a counter-clockwise traversal of the edges of the polygons containing the origin. The algorithm begins with an arbitrary edge e which is either partly or fully visible to the origin. Every time an intersection point is encountered, the edge that makes the sharpest counter-clockwise turn with respect to the origin is selected. The algorithm terminates when it returns to the starting point on e.

4.3 Hypothesis Elimination

A set R of points is picked randomly from the connected intersecting region of the overlay arrangement surrounding the origin (see Figure 3). R is then evaluated to see if any of the points contained in R yield new information that could help to disambiguate the robot's initial location. Our decision strategy weights the utility of a potential destination point with its distance from the robot's initial location. Before we proceed we will define some terms:

Definition 2 (Random Point).
A random point *refers to a location in the connected component OI of the overlay intersection region containing the origin for a given P, H, and origin, chosen by randomly sampling the interior of OI according to a uniform distribution.*

Definition 3 (Useful Point).
A random point q is termed useful *if sensing at this location is guaranteed to yield new information that distinguishes amongst the different hypothetical locations. In other words, the $Vis(q)$ with respect to the different hypotheses are not all congruent under translation.*

For each random point picked, $r \in R$, a value function $Value(r) = info/distance_{OI}$ is computed, where $info$ is the expected number of hypotheses that could be eliminated at r, assuming all the hypothesized initial locations are equally likely, and $distance_{OI}$ is the shortest path trajectory,

constrained to lie within OI, from the robot's initial location at the origin of the overlay to r. We calculate $info$ for a point r as follows:

We assume that all hypotheses are equally likely. We say two hypotheses h_i and h_j are equivalent at r if $Vis(h_i, r)$ is congruent to $Vis(h_j, r)$ and has the same orientation. $Vis(h_i, r)$ refers to the visibility data computed at a point z such that the relative position of z with respect to the hypothetical location p_i is equivalent to the relative position of r with respect to the overlay origin. If there exist b equivalence classes of hypotheses at r of sizes $s_1, s_2, ..., s_b$ respectively, where the total number of hypotheses $k = s_1 + s_2 + ... + s_b$, then

$info(r) = (s_1/k)(k - s_1) + (s_2/k)(k - s_2) + ... + (s_b/k)(k - s_b)$.

The robot is moved to the random point r' in OI with the highest non-zero value of $Value(r')$. Those hypotheses h_i where $Vis(h_i, r')$ does not match the visibility data sensed by the robot at its new location are ruled out.

4.4 Common Overlay Localization Algorithm

We can now present the *common overlay localization (COL)* algorithm. Given an input polygonal environment P and a robot placed at an unknown initial location in P, the COL algorithm proceeds as follows:

1. Sense visibility data W from the robot's current unknown initial location.
2. Generate the set of all hypothetical locations H in the environment P that match the visibility data sensed W.
3. Choose an arbitrary hypothetical location in H as the origin.
4. Construct an overlay arrangement centered on the origin.
5. Compute the connected overlay intersection component containing the origin, OI.
6. Randomly choose a predetermined number of points within OI.
7. For each random point picked, r, compute the value function $Value(r) = info/distance_{OI}$.
8. Observe that at each overlay intersection, there is latent information to be gained that is guaranteed to eliminate some hypotheses. Therefore, if none of the random points yield non-zero $info$, then the number of random points required is increased and chosen all over again within the current overlay intersection area OI. Steps 6, 7 and 8 are repeated for a predetermined number of trials [3].
9. The robot moves to the random point r' in the overlay with the highest non-zero value of $Value(r')$.
10. Now, eliminate hypotheses by comparing visibility data sensed by the robot at r' with the visibility data computed at all the equivalent random points corresponding to all the active hypotheses.

[3] In our implementation, we terminate the algorithm if no useful points are obtained after this predetermined number of trials. Hence we proceed to the next step only if useful points exist.

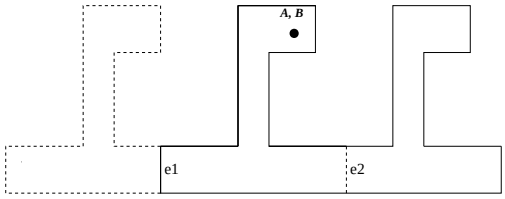

Fig. 4. e1 and e2 are internal edges.

11. Let us call the set of eliminated hypotheses E. We repeat the overlay arrangement with the reduced set of hypotheses $H - E$. Steps 3-10 are repeated until only 1 hypothesis, corresponding to the true initial location of the robot, is left in $H - E$.

4.5 Useful Region Localization Algorithm

The *useful region localization (URL)* algorithm differs from the COL algorithm with respect to the region where random points are chosen. Recall that the COL algorithm randomly chooses points within the overlay intersection area OI. The COL algorithm then selects only those random points that are expected to yield new information. It turns out that we can do even better. In fact, we can determine precisely the portions of OI where any random point chosen is guaranteed to provide new information.

Let us first observe that in large sections of OI, the visibility cells provide zero information gain for unambiguous localization. We must look to the boundaries of OI which is where the area common to all hypotheses ends and the"geography" changes. But the robot can spot this change from a distance as it approaches the boundaries. We refer to these boundaries as *internal edges*, which we define as follows:

Definition 4 (Internal Edge).
An internal edge *of an overlay intersection area OI is defined as an edge (one of many) that separates the inside of OI from other parts of the overlay arrangement, as opposed to those edges of OI that pertain to the outer silhouette of the overlay arrangement which separates the inside of OI from the rest of the 2D plane (see Figure 4).*

Before we proceed to the description of the URL algorithm, we will examine the notion of *weak* visibility from an edge. A set of points Q is said to be *weakly visible* from an edge e if for each point $q \in Q$ there exists a point $z \in e$ such that q and z are visible (z may depend on q) [9].

Definition 5 (Weak Visibility Polygon).
Given a polygon P the weak visibility polygon $W(e)$ *of an edge $e \in P$ is defined as the set of all points $y \in P$ that are visible from some point on e (see Figure 5).*

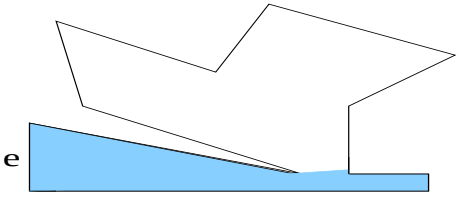

Fig. 5. Shaded region represents the weak visibility polygon of edge e.

Once we have determined the set of internal edges of OI, the useful portions U of OI can be computed using the following procedure:

- Let us call the set of internal edges of OI, B. For each edge in B, compute its weak visibility polygon within OI. The union of all such weak visibility polygons should give us a region or set of disjoint regions U where U is a subset of OI. We claim that any random point chosen in U provides non-zero information whereas any random point chosen in OI-U provides zero information.

Figure 6 depicts the useful region of polygon P. Steps 1-5 of the URL algorithm remain the same as in the COL algorithm. Following these steps, the URL algorithm proceeds as described below:

6. Compute the useful region U of OI.
7. Randomly choose a predetermined number of points within U (Figure 6).
8. For each random point picked, r, compute the value function $Value(r) = info/distance_{OI}$.
9. The robot moves to the random point r' in the overlay with the highest non-zero value of $Value(r')$. Note that we are guaranteed that all the random points chosen provide non-zero information for hypothesis elimination. As a result, we do not need to choose more random points repeatedly as is done in the COL algorithm.
10. Now, eliminate hypotheses by comparing visibility data sensed by the robot at r' with the visibility data computed at all the equivalent random points corresponding to all the active hypotheses.
11. Let us call the set of eliminated hypotheses E. We repeat the overlay arrangement with the reduced set of hypotheses $H - E$. Steps 3-10 are repeated until only 1 hypothesis, corresponding to the true initial location of the robot, is left in $H - E$.

It must be noted that once we have computed the useful region U, we need choose only one random point lying within U at each stage in order to guarantee that we eventually localize successfully (we could even just move the robot to the closest vertex or edge of U). A larger number of random points only serves to improve the performance of the algorithm by eliminating several hypotheses in one shot or by reducing the distance traveled by the robot.

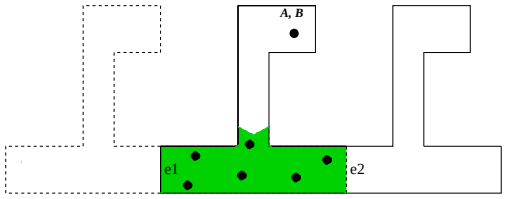

Fig. 6. Shaded area represents the useful region. Random points are chosen within this region.

Unlike the COL algorithm, the URL algorithm has computable and finite time bound. Consider a situation where the useful region comprises a small fraction of the entire overlay intersection area OI (this need not mean that the useful region is in itself a small area where a robot can not navigate - simply that it is relatively small compared to the entire overlay intersection area). In such cases, choosing points randomly from the entire overlay intersection region may yield a useful point only with a large number of random points and after several trials. The advantage of the URL algorithm is that we can instantly access the useful portions of OI and hence the useful points [4].

4.6 Complexity Analysis

In this section, we provide an estimate of the overall time complexity for the two localization algorithms. The number of initial hypothetical locations k is bounded above by the number of reflex vertices in the polygonal environment [5]. Let f be the number of active hypotheses remaining ($f \leq k$). Let n denote the number of vertices in the map polygon P.

Hypothesis generation takes time $O(mn)$ where m is the number of vertices in the visibility data W. Calculating the overlay intersection area takes time $O(fn \log n)$. Observe that as hypotheses are ruled out the value of f will decrease. If we rule out only one hypothesis per intersection then the overlay intersection is computed a maximum of k times. It takes $O(fn)$ time per random point in order to calculate the information gain at any random point. Computing the shortest path distance of a random point from the initial location amounts to $O(n \log n)$ time.

Common Overlay Localization Algorithm

At each intersection let us say that we select X random points on average. Let only Y be the number of useful points where $Y \leq X$. Assuming that we only eliminate one hypothesis at each overlay intersection, we will have to calculate the overlay intersection area, compute and compare visibility data of all the

[4] A proof of correctness for the URL algorithm is presented in [10].
[5] For a proof see [6].

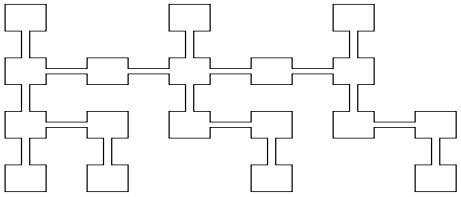

Fig. 7. A simulated office environment.

random points, and calculate the shortest path distance of the useful random points, a total of k times. Therefore we estimate the overall time complexity of the COL algorithm to be $O(mn) + k(O(fn \log n) + O(Xfn) + O(Yn \log n))$. Since $k = O(n)$ and $f = O(k)$, a looser but simpler bound for the overall time complexity of the COL algorithm is obtained: $O(n^3 \log n + Xn^2)$.

Useful Region Localization Algorithm

The maximum number of internal edges possible at any overlay intersection is $O(fn)$. Let Y be the number of useful random points. Computing the weak visibility of an edge takes $O(n \log n)$ time [9]. As a result, the total cost of computing the useful region is $O(fn^2 \log n)$. Therefore we estimate the overall time complexity of the URL algorithm to be $O(mn) + k(O(fn \log n) + O(fn^2 \log n) + O(Yfn) + O(Yn \log n))$ which reduces to $O(n^4 \log n + Yn^3)$.

5 Experimental Results

Experiments were carried out on random environments in order to obtain an empirical measure of the average path length for different numbers of random points and with respect to the expected optimum path length. We also compare the performance of our strategy with competing strategies. In order to test the localization algorithms, we used random simulated "office" environments that were generated algorithmically (see Figure 7).

We generated 73 simulated office environments with an average of 400 vertices each. For each environment an initial robot location was randomly selected. We then ran the COL algorithm with these environments and their respective initial locations for a series of different quantities of random points. The objective was to measure the average distance traveled by the robot as the number of random points was increased and to compare these average path lengths to the estimated optimum result. The number of random points was varied from 20 to 1000 and the average path length obtained for 1000 points was used as the estimated optimum result. Figure 8 shows the error margin of the average path length with respect to the estimated optimum path length plotted against the number of random points. Our results indicate that the average path length gets significantly shorter initially as the number of

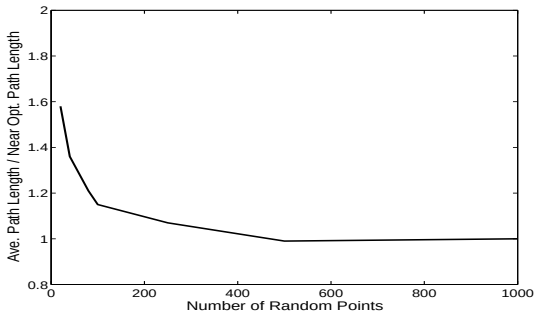

Fig. 8. Ratio of path length obtained with COL algorithm with respect to the expected optimum path length based on 73 trials. Near-optimal path length is achieved with a relatively small number of points.

random points is increased. Eventually, the incremental reduction in path length decreases and the path length settles at the near-optimum value [6].

We ran experiments to compare the performance of the *weighted decision strategy* used in the COL algorithm (which we will refer to as the weighted strategy) versus its greedy version, where the robot is directed to move to the nearest location to its initial location that provides any non-zero information. This greedy alternative *approximates* the greedy technique of [3] but is not equivalent to it since we require an appropriate minimum number of points in order to adequately cover all the visibility cells in the overlay intersection region. We generated 20 simulated office environments with an average of 400 vertices each. For each of the 20 environments, 3 initial locations were randomly selected. We then ran both weighted and greedy strategies with each of the 3 initial locations for each environment for a series of different quantities of random points, repeating each quantity twice to balance out any abnormal distributions. The total number of experimental trials was, therefore, $20 \times 3 \times 2 = 120$ trials [7]. Figure 9 shows the average path length for weighted and greedy strategies plotted against the number of random points. Even with very large numbers of random points (see 1000 points case), where we may reasonably assume an adequate sampling of most of the visibility cells in the overlay intersection region, on average the weighted strategy still outperforms its greedy counterpart.

Theoretically, the greedy localization strategy has been shown to have the best possible worst case bound on the distance traveled by the robot [3]. In our weighted strategy as well as its greedy variant, it is possible that a tiny visibility cell which might hold the key to an optimally short path was never sampled. This is a theoretical disadvantage but a practical advantage as the

[6] Recall that computing the optimum path is NP-hard.

[7] This set of trials will be used repeatedly for the experiments described in the remainder of this paper.

Fig. 9. Performance of weighted decision strategy of COL algorithm vs. greedy strategy based on 120 trials. Weighted strategy gives shorter path lengths on average.

probability that the robot is directed to visit an arbitrarily tiny, and hence inaccessible, visibility cell is small.

A set of experiments, involving 120 trials, was performed where the weighted decision strategy of the COL algorithm was compared with a traditional heuristic used in many localization papers. The heuristic directs the robot to simply visit the closest point from its *current* location which provides any non-zero information. Figure 10 shows the average path length for our weighted strategy and the traditional heuristic plotted against the number of random points. The weighted strategy produces shorter path lengths than the heuristic.

Experiments were carried out to evaluate the performance of the URL algorithm with respect to different numbers of useful random points. We performed the experiments over 120 trials as described above. Figure 11 depicts the average path length obtained for number of useful random points ranging from 1 to 500. Since we are choosing points directly from the useful region, the algorithm is able to effectively localize the robot with just 1 random point, although the path length is understandably high in that case. We observe that path length values for the URL algorithm are somewhat higher than those obtained for the same number of points when running the COL algorithm. This is because the COL algorithm chooses a more dense sampling of random points each time none of the existing points prove useful. In particular, the overlay intersection areas get larger as hypotheses are eliminated thereby requiring a more dense sampling. On the other hand, the URL algorithm adheres to exactly the same number of points initially specified, regardless of the size of the subsequent overlay intersection regions.

Figures 12 and 13 depict the localization path taken by the robot for a staircase-like environment and a simulated office environment respectively, using the COL algorithm. The shaded area in these figures represents the overlay intersection region OI. The square black points labeled $H0, H1, ..., Hk$ indicate the different hypothetical locations. The small round dots scattered across the shaded region represent the random points.

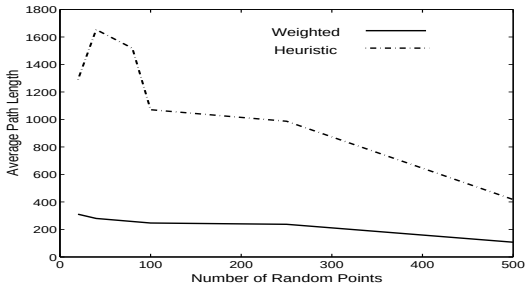

Fig. 10. Performance of weighted decision strategy of COL algorithm vs. traditional heuristic based on 120 trials. Weighted strategy gives shorter path lengths on average.

6 Discussion and Conclusions

In this paper we have presented a new algorithmic formulation of minimum distance localization based on randomized sampling and determined its complexity. We demonstrate that although minimum distance localization is a hard problem, a good approximation can be achieved in most cases with limited computation.

The experimental results show that for the ensemble of environments we have evaluated, our algorithms are effective. The performance of our algorithms improves rapidly with the number of random sample points used. However, the incremental improvement decreases as the number of samples increases, so that a fairly limited number of samples typically is sufficient to obtain a near-optimal length localization trajectory.

In the case of the URL algorithm, we require as little as 1 random point in order to unambiguously localize the robot. In situations where the useful region comprises a small fraction of the entire overlay intersection area, the URL algorithm seems to be the better choice (than the COL algorithm). Choosing points randomly from the entire overlay intersection region may yield a useful point only with a relatively large number of points. The advantage of the URL algorithm is that we can instantly access the useful portions of the overlay intersection and hence the useful points.

In addition, experimental results indicate that on average, our weighted decision strategy is clearly superior to alternative greedy strategies which simply visit the nearest points to the robot's current, or even starting, location, that offer any information at all. The strategy of picking points stochastically leads to a visibility cell having a probability of being visited that is proportional to its area. This implies that large cells, that are accessible to the robot, will be chosen for visits more frequently than small ones. This in itself makes our algorithm more feasible for real implementation than its predecessors.

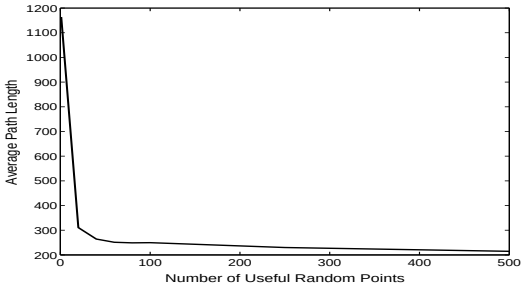

Fig. 11. Performance of URL algorithm based on 120 trials. More dense sampling of useful random points produces rapid reduction in path length.

While the validity of these results is formally limited to the class of environments presented here, there is no reason why these conclusions should not generalize to broader classes of environments. Our algorithms appear to be appropriate to use in contexts where the sensor is more limited, given suitable modifications to the definition of a landmark. This suggests a natural extension to an even more realistic context.

References

1. R.G. Brown and B.R. Donald, *Mobile robot self-localization without explicit landmarks*, Algorithmica **26** (2000), no. 3/4, 515–559.
2. G. Dudek and M. Jenkin, *Computational principles of mobile robotics*, Cambridge University Press, 2000.
3. G. Dudek, K. Romanik, and S. Whitesides, *Localizing a robot with minimum travel*, SIAM J. Computing **27** (1998), no. 2, 583–604.
4. D. Fox, W. Burgard, and S. Thrun, *Active markov localization for mobile robots*, Robotics and Autonomous Systems **25** (1998), 195–207.
5. H.H. Gonzalez-Banos and J.C. Latombe, *Planning robot motions for range-image acquisition and automatic 3d model construction*, In Proc. AAAI Fall Symposium Series, 1998.
6. L. Guibas, R. Motwani, and P. Raghavan, *The robot localization problem*, SIAM J. Computing **26** (1997), no. 4, 1120–1138.
7. J. Kleinberg, *The localization problem for mobile robots*, In Proc. 35th IEEE Conference on Foundations of Computer Science (Santa Fe, NM), IEEE Computer Society Press, 1994, pp. 521–533.
8. S. Schuierer, *Sensing, modelling and planning*, Intelligent Robots, ch. Efficient robot self-Localization in simple polygons, pp. 129–146, World Scientific Publ., 1996.
9. G.T. Toussaint, *A linear-time algorithm for solving the strong hidden-line problem in a simple polygon*, Pattern Recognition Letters **4** (1986), 449–451.
10. M. Rao, *A Randomized Approach to Minimum Distance Localization*, MSc. thesis, Dept. of Computer Science, McGill University, Montreal, Canada, 2004.

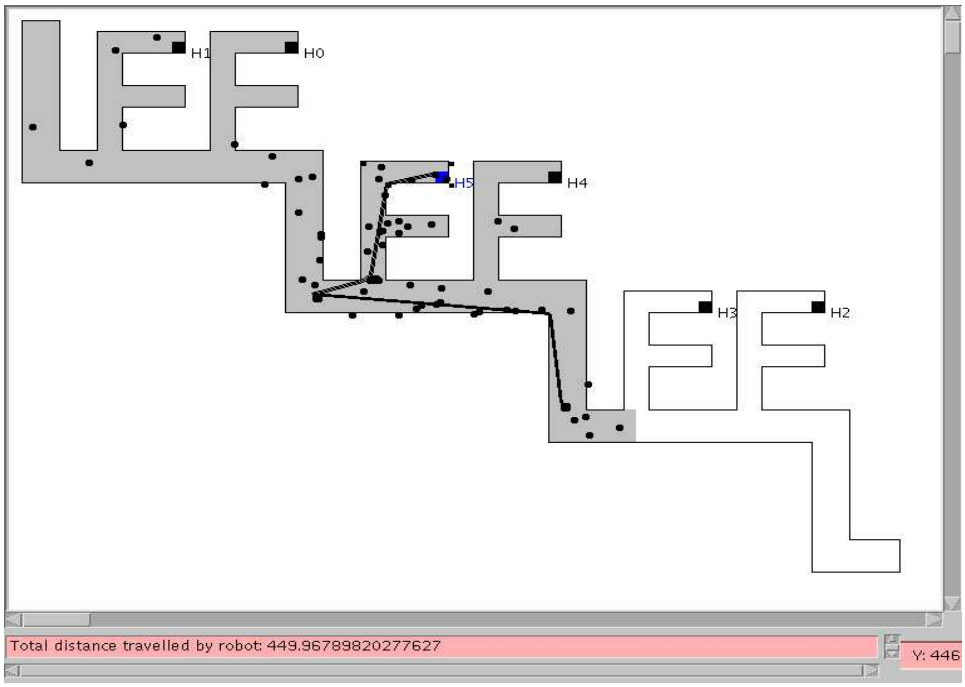

Fig. 12. Localization trajectory in staircase environment.

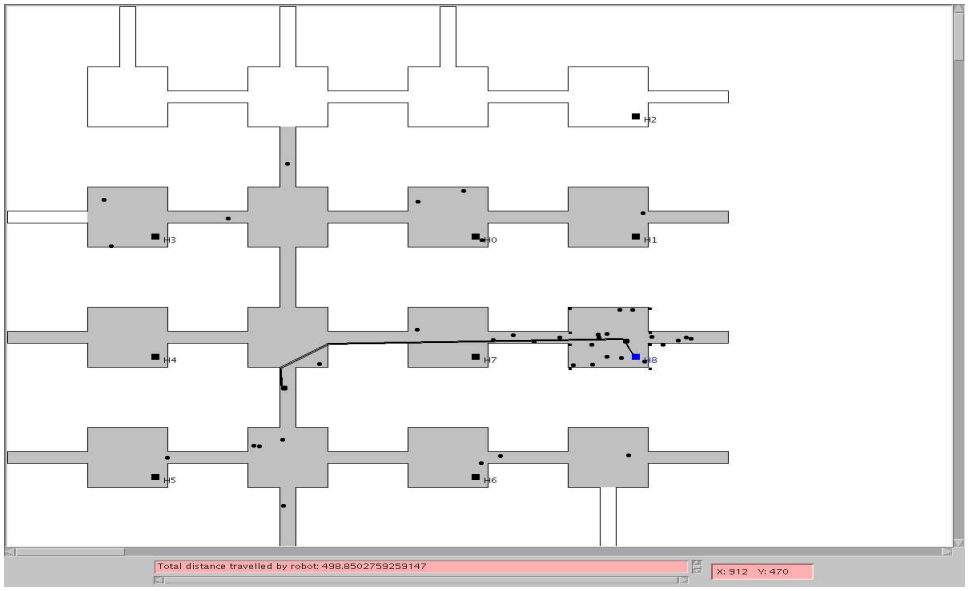

Fig. 13. Localization trajectory in simulated office environment.