Local Exploration: Online Algorithms and a Probabilistic Framework

Volkan Isler, Sampath Kannan, and Kostas Daniilidis

Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, 19104
{isleri,kannan,kostas}@cis.upenn.edu

***Abstract*— Mapping an environment with an imaging sensor becomes very challenging if the environment to be mapped is unknown and has to be explored. Exploration involves the planning of views so that the entire environment is covered. The majority of implemented mapping systems use a heuristic planning while theoretical approaches regard only the traveled distance as cost. However, practical range acquisition systems spend a considerable amount of time for acquisition. In this paper, we address the problem of minimizing the cost of looking around a corner, involving the time spent in traveling as well as the time spent for reconstruction. Such a local exploration can be used as a subroutine for global algorithms. We prove competitive ratios for two online algorithms. Then, we provide two representations of local exploration as a Markov Decision Process and apply a known policy iteration algorithm. Simulation results show that for some distributions the probabilistic approach outperforms deterministic strategies.**

I. Introduction

In robotics, mapping is the recovery of environmental layouts from measurements obtained by sensors mounted on mobile robots. Mapping is a very active research area and a recent survey of the state of the art can be found in [18]. The task becomes very challenging when the environment is unknown and when robot pose has to be estimated from the same measurements used for the mapping (simultaneous localization and mapping). An additional challenge in unknown environments is the issue of visual coverage or better known as visual exploration. We emphasize the visual aspect of coverage (as in [10], [17]) as opposed to area coverage meant either as producing a roadmap [7] or sweeping of space, for example in the case of vacuum cleaners or landmine detection. Usually many of the general exploration algorithms produce a redundant visual coverage and are thus inefficient if visual coverage and mapping is the main purpose.

Visual exploration is a planning problem facing the issues of completeness (see everything) and optimality (in minimal time). Usually optimality is estimated in terms of traveled distance but such an estimation assumes that range acquisition can be performed in minimal time and on the fly. This is not the case with laser scanners where the robot has first to stay stationary and obtain a range map before deciding where to go or for stereo vision systems where computation cannot be pipelined with image grabbing.

The novelty of this paper is in addressing the problem of spending time in range acquisition which has not been accounted for in previous exploration approaches. The number of reconstructions is implicitly considered in view planning and in particular in the best next view problem [15], however, without any optimality claims.

In this paper we consider the specific problem of finding a view planning strategy so that an occluded edge becomes visible under the minimal time spent for reconstruction and traveling. Our algorithm can be used as a subroutine by a greedy planner (for e.g. as in [13]), which tries to see the "next" invisible edge of a polygonal environment in order to reduce the total time of reconstruction and traveling.

The closest related algorithms are the competitive exploration algorithms we will refer to in the next subsection. The cost of reconstruction is addressed by Rekleitis et al. [14] who use two robots for visual exploration where one robot employs the function of range acquisition while the other remains in line of sight and its measurement plan the next view of the former robot. Zlot et al. [19] present a multi-robot approach for exploration trying to maximize information gain with minimizing incurring costs. Burgard et al. [6] assign a new target point for each of a group of robots so that the cost of reaching these points is minimized and the amount of already explored area is simultaneously maximized.

A. Online algorithms and competitive analysis

Traditional algorithms typically operate on the entire input. In online problems [2] the input is not known in advance but presented to the online algorithm during its operation instead. One way of measuring the performance of online algorithms is competitive analysis [5]. In competitive analysis, we compare the performance of an online algorithm against the performance of the optimal offline algorithm and consider the worst case ratio. Let $cost_A(\sigma)$ be the cost incurred by an online algorithm A on the input sequence σ. Let OPT be the optimal offline algorithm and let $cost_{OPT}(\sigma)$ be the cost incurred by the optimal offline algorithm on input σ. We say that the online algorithm A is *c-competitive*, if there exits a constant b such that on every input sequence σ,

$$cost_A(\sigma) \leq c \cdot cost_{OPT}(\sigma) + b$$

The *competitive ratio* is the infimum over c such that A is c-competitive. We say that an algorithm is competitive, if it has a constant competitive ratio. In robotics, competitive analysis has been used for various navigation problems as a measure of efficiency [4], [9], [1], [12], [8], [11]. In the context of exploration, the competitive ratio gives us the worst case deviation of the cost of an exploration algorithm from the cost incurred by a robot who has a prior model of the environment and still wants to build a map.

B. Competitive analysis in robot exploration

A 2-competitive algorithm for rectilinear polygons with bounded number of obstacles has been presented in [8]. For simple polygons without obstacles, a 26.5-competitive algorithm has recently been proposed [11]. For polygons with an arbitrary number of obstacles, it has been shown that there is no competitive strategy [1]. For the local problem of how to look around a corner, which is addressed in this paper, a 1.21-competitive algorithm has been presented [12].

All above algorithms make the continuous visibility assumption that the robot can continuously acquire a 3D view of the environment without any stop or cost for this acquisition. This assumption is violated for range scanners where the robot has to stop and acquire the locally visible 3D-view. It does not apply for omnidirectional visual stereo reconstruction either, because current acquisition times do not allow on the fly computation: the robot can only decide where to move after acquiring the map.

C. New problem statement

In this paper, we address local exploration strategies which can arise in global exploration strategies. We make the following assumptions:

- The 3D-environment consists of vertical edges and walls and thus can be modeled as a polygon in the flatland.
- We assume that the robot can localize itself with respect to an acquired view and that it can register these views in the same coordinate system. In this paper, we start with the case of no uncertainty in the robot's position estimate.
- We assume that the robot has an omnidirectional range acquisition system, which means no restrictions in the field of view.
- We assume that the robot does not move during range acquisition.
- We assume that the circle defined by robot's current position and the vertex adjacent to the edge to be explored (figure 1) is free of obstacles.

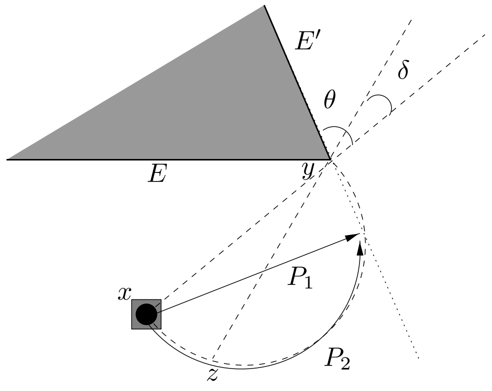

Fig. 1. The problem description: The robot, located at x, can see the edge E but not E', the next edge. E' makes an angle of θ with the line passing through x and the corner y. The optimal offline strategy is P_1, going directly to the extension of E' when $\theta < \frac{\pi}{2}$ and to go directly to y otherwise. When robot has continuous vision, by following P_2 along the circle whose diameter is $\overline{xy}$, the robot guarantees a competitive ratio of $\frac{\pi}{2}$.

In this setup, we consider a robot, located at x, seeing the edge E but not E', the next edge (see figure 1). Edge E' makes an angle of θ with the line passing through x and the corner y. The optimal offline strategy is to follow path P_1, going directly to the extension of E' when $\theta < \frac{\pi}{2}$ and to go directly to y otherwise. In the online setting it is not possible to follow P_1, because its orientation depends on E' which has not been seen by the robot yet. When the robot has continuous vision, by following P_2 along the circle whose diameter is $\overline{xy}$, the robot guarantees a competitive ratio of $\frac{\pi}{2}$. This strategy was used in [11] as a part of the global exploration strategy assuming on the fly acquisition.

Here, we introduce a new cost measure for the time it takes to see the next occluded edge as the sum of the time spent in reconstructions plus the time spent in traveling:

$$cost_A(\sigma) = \tau N + \frac{d}{v}, \tag{1.1}$$

where τ is the time it takes to make a reconstruction, N is the number of reconstructions made until next edge is seen, d is the distance traveled, and v is the velocity of the robot. The input σ consists of robot's position x, the position of the corner vertex y, and the angle θ the next edge makes with the robot's line of sight (see figure 1). Note that our cost model assumes constant velocity, however it is possible to incorporate more complicated dynamics into equation 1.1.

The contribution of this paper is two-fold:

- In a deterministic set-up with no knowledge about the occluded edge, we present two competitive strategies.
- Assuming a belief about the occluded edge, we propose two formalizations in terms of a Markov Decision Process (MDP) and solve for optimal policies that maximize the overall expected reward.

In simulations, we compare the four algorithms and we find out that the MDP policies outperform the deterministic algorithms when the beliefs are close to the reality. The paper is written in the just described order: competitive algorithms, MDP framework, and experimental analysis.

II. Competitive Algorithms

Let x be robot's position, y be the corner, D be the distance from the robot's current position to the corner, v be the speed of the robot, and τ be the time it takes to make a reconstruction. Let $\epsilon\frac{D}{v} = \tau$. That is, the time it takes to make a reconstruction is ϵ times the time it takes to reach the corner. Let $t_{OPT} > 0$ be the time it takes the optimal algorithm to reach the point it can see the next edge (traversing P_1 in Fig. 1).

If $\epsilon \geq 1$, then the robot goes straight to the corner. Since $\frac{D}{v} \leq \tau$, the competitive ratio becomes

$$\frac{\tau + \frac{D}{v}}{\tau + t_{OPT}} \leq \frac{2\tau}{\tau + t_{OPT}} \leq 2.$$

Otherwise, we propose two algorithms as described in Table I.

UNIREC(τ, v, x, y)	EXPREC(τ, v, x, y)
$D \leftarrow dist(x, y)$	$D \leftarrow dist(x, y)$
$\mathcal{C} \leftarrow circle(x, y)$	$\mathcal{C} \leftarrow circle(x, y)$
$\delta \leftarrow \frac{\tau v}{D}$	$\delta \leftarrow \frac{\tau v}{D}$
If $\delta > 1$ go to x	If $\delta > 1$ go to x
Otherwise	Otherwise
$i = 1$	$i = 1$
Until the next edge is seen	Until the next edge is seen
Visit $i\delta$	Visit $i\delta$
Reconstruct	*Reconstruct*
$i \leftarrow i + 1$	$i \leftarrow 2i$

TABLE I

THE INPUT x IS THE ROBOT'S POSITION, y IS THE LOCATION OF THE CORNER, v IS ROBOT'S SPEED, AND τ IS THE TIME IT TAKES TO MAKE A RECONSTRUCTION. THE COMMAND *Reconstruct* DENOTES THE OPERATION OF AN OMNIDIRECTIONAL RANGE ACQUISITION AND $circle(x, y)$ IS THE CIRCLE THAT PASSES THROUGH x AND y AND HAS A DIAMETER $dist(x, y)$. ALGORITHM UNIREC HAS A COMPETITIVE RATIO OF π AND ALGORITHM EXPREC HAS A COMPETITIVE RATIO OF 2.23.

A. Algorithm UNIREC

Let $\mathcal{C}$ be the circle whose diameter is the line segment that joins the robot to the corner (i.e x to y). Suppose in Fig. 1, that during time τ the robot travels to position z on $\mathcal{C}$ without leaving the circle. Let $\delta = \angle xyz = \frac{\tau v}{D}$. Note that $\epsilon = \delta$. The robot will go to the points on $\mathcal{C}$ defined by $\delta, 2\delta, 3\delta, \ldots$ until it sees the next edge without leaving the circle [1]. Let $\theta \in [0, \frac{\pi}{2}]$ be the actual angle (Fig. 1 between the edge and the line that passes through the robot's position and the corner). The competitive ratio of this algorithm reads:

$$C = \frac{\lceil\frac{\theta}{\delta}\rceil\tau + \lceil\frac{\theta}{\theta}\rceil\theta\frac{D}{v}}{\tau + \frac{D}{v}\sin\theta}$$

Since $\frac{D}{v} = \frac{\tau}{\delta}$ we obtain

$$C = \frac{\lceil\frac{\theta}{\delta}\rceil\tau + \lceil\frac{\theta}{\delta}\rceil\delta\frac{\tau}{\delta}}{\tau + \frac{\tau}{\delta}\sin\theta} = \frac{2\lceil\frac{\theta}{\delta}\rceil}{1 + \frac{\sin\theta}{\delta}} \leq \frac{2(\frac{\theta}{\delta} + 1)}{1 + \frac{\sin\theta}{\delta}}$$

which is increasing with θ. Hence, the worst case is achieved when $\theta = \pi/2$:

$$C \leq \frac{2(\frac{\pi}{2\delta} + 1)}{1 + \frac{1}{\delta}} = \frac{2\delta + \pi}{\delta + 1}$$

Since $\delta < 1$, the worst case is achieved as $\delta \to 0$ and the ratio becomes π.

B. Algorithm EXPREC

It is possible to improve this ratio by modifying the strategy as follows: Instead of visiting $\delta, 2\delta, 3\delta, \ldots$, the robot increases exponentially its steps and visits $\delta, 2\delta, 4\delta, \ldots, 2^i\delta$. Note that during the i^{th}-step robot traverses an angle of $2^{i-1}\delta$ and the total angle traversed so far is $(2^i - 1)\delta$. If θ is the actual angle, the robot sees the next edge as soon as it takes $i = \lceil\log(\frac{\theta}{\delta} + 1)\rceil$ steps. The competitive ratio reads:

$$C = \frac{i\tau + \delta(2^i - 1)\frac{D}{v}}{\tau + \frac{D}{v}\sin\theta}$$

The worst case of the ratio of EXPREC is thus 2.2214. We present the details of this straightforward but lengthy derivation in the appendix.

III. Probabilistic framework

In most environments, we expect that the robot has some expectation about the angles formed by vertices in polygonal environments. For example, most angles in man-made environments are rectilinear or in case of doors 180 degrees. In this section, we present a framework that allows us to represent robot's belief about the environment as a probability distribution and show how to solve for optimal strategies when such beliefs are available.

A finite state Markov Decision Process (MDP) is given by a finite set of states S, a finite set of actions A,

[1]The reader may wonder why we do not take the short cuts instead, which means compute δ and go straight to the point $(D\cos\delta, D\sin\delta)$ and continue with updating $D \leftarrow D\cos\delta$. Even though this might perform better for some values of τ, it does not improve the competitive ratio for small δ: $D\cos\delta \approx D$ and $D\sin\delta \approx D\delta$

transition probabilities $P(r|s,a)$ of arriving at state r when action a is taken from state s, and rewards $R^a_{s,r}$ from arriving at state r from state s via action a. A policy π is a function that takes a state-action pair (s,a) and returns a real number in [0,1], indicating the probability of taking action a when in state s. An optimal policy is a policy whose expected return from each state is greater than any other policy for all states. Given a finite MDP it is possible to find an optimal policy using dynamic programming or its variants such as Policy Iteration. A comprehensive introduction to MDP can be found in [16], [3].

Suppose we have the distribution $P^\theta(\theta)$ for the distribution of the corner angles. For example, one can express the belief that the environment is rectilinear by choosing $P^\theta(\theta)$ to be a truncated gaussian with mean 90 degrees and a variance representing the uncertainty of this belief. Another possibility is to keep the histogram of the angles already observed during the exploration and to use this histogram as an approximation for $P^\theta(\theta)$. Yet another possibility is to use Monte Carlo Methods [16] for reinforcement learning to incorporate the learning of $P^\theta(\theta)$ into the exploration process. Even though obtaining $P^\theta(\theta)$ is an interesting problem on its own, from now on we assume that it is given as an input. One way to model the edge exploration problem is to discretize the circle whose diameter is the line segment joining the robot and the vertex using a resolution parameter δ. Let $n = \frac{\pi}{2\delta}$ and let us double use the notation $\delta, 2\delta, 3\delta, \dots n\delta$ for both the stops on the circle as well as the angles whose apex is at the vertex.

An MDP model, we will call MDP1, is presented in figure 2. State s_i represents the state of the robot when it is located at $i\delta$ and has not made a reconstruction yet. At each s_i it can either decide to move to s_{i+1} or make a reconstruction. When it makes a reconstruction it either sees the next edge in which case it goes to the state $\mathcal{F}$ and remains there or cannot see it yet. The latter case is represented by the state s'_i. From s'_i the only reasonable action is to move. Note that we chose to discretize the circle defined by the robots location and the corner, instead of discretizing the whole plane. The advantage of this approach is the drastic reduction in the number of states which means a reduction in the memory requirements and running time of the algorithm.

The actions are Rec and Mov for reconstruct and move respectively. The transition probabilities are determined by the distribution $P^\theta(\theta)$:

$$\begin{aligned} P(\mathcal{F}|s_i, Rec) &= P^\theta(\theta \le i\delta) \\ P(s'_i|s_i, Rec) &= P^\theta(\theta > i\delta) \\ P(s_{i+1}|s_i, Mov) &= 1 \\ P(s_{i+1}|s'_i, Mov) &= 1 \end{aligned}$$

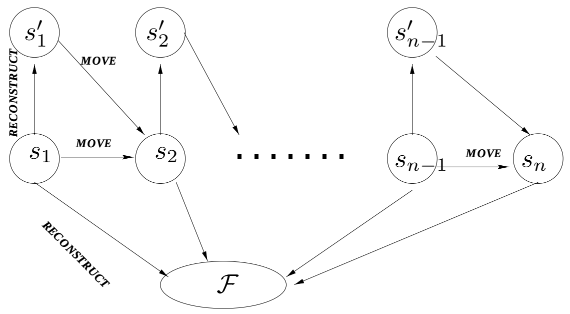

Fig. 2. **MDP1** has $2n$ states where $n = \frac{\pi}{2\delta}$ that depends on the sampling parameter δ and $\mathcal{F}$ is the final state. Being in s'_i (resp. s_i) means that the robot is at δi and has just (resp. not) made a reconstruction.

All other probabilities are zero. Note that even though we assumed that the robot has complete control of its motion by letting $P(s_{i+1,j}|s_{i,j}, Mov) = 1$, one can easily incorporate uncertainty in motion using an appropriate uncertainty model.

The rewards, $R^a_{s_i,s_j}$, represent the immediate reward received upon arriving state s_j from state s_i as a result of action a. Since we are dealing with costs, we use negative costs as rewards we want to maximize.

$$\begin{aligned} R^{Mov}_{s_i,s_{i+1}} &= -\frac{\delta D}{v} \\ R^{Mov}_{s'_i,s_{i+1}} &= -\frac{\delta D}{v} \\ R^{Rec}_{s_i,s'_i} &= -\tau \\ R^{Rec}_{s_i,\mathcal{F}} &= -\tau \end{aligned}$$

Given a distribution P^θ, we compute the optimal policy that maximizes the expected reward using the well known policy iteration algorithm [16, pp98]. Policy iteration is known for its fast convergence properties in practice and this was indeed the case for our problem. For MDP1, we observed that the optimal policies move until enough probability is accumulated and start reconstructing afterwards. For example, if $P^\theta = \mathcal{N}(\mu, \sigma)$, the optimal algorithm turns out to move an angle of $\mu + \sigma$ and then to reconstruct at each step afterwards.

It is possible to obtain a better performance by remembering the last reconstruction made. Let s_{ij} represent the information that the robot is standing at $i\delta$ and the last reconstruction it made was at $j\delta$. Figure 3 illustrates the transitions for state $s_{i,j}$. The transition probabilities and rewards for this new MDP, which we call MDP2, are given by:

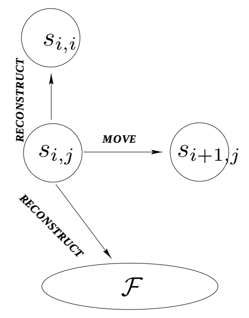

Fig. 3. **MDP2** has $\frac{n(n+1)}{2}$ states where $n = \frac{\pi}{2\delta}$ that depends on the sampling parameter δ. $\mathcal{F}$ is the final state. Being in state $s_{i,j}$ means that the robot is at $i\delta$ on the circle and the last reconstruction was at $j\delta$.

$$\begin{aligned}
P(\mathcal{F}|s_{i,j}, Rec) &= P^{\theta}(j\delta \leq \theta \leq i\delta) \\
P(s_{i,i}|s_{i,j}, Rec) &= 1 - P(\mathcal{F}|s_i, Rec) \\
P(s_{i+1,j}|s_{i,j}, Mov) &= 1
\end{aligned}$$

All other probabilities are zero.

$$\begin{aligned}
R^{Mov}_{s_{i,j},s_{i+1,j}} &= -\frac{\delta D}{v} \\
R^{Rec}s_{ij}, s_{i,i} &= -\tau \\
R^{Rec}s_{i,j}, \mathcal{F} &= -\tau
\end{aligned}$$

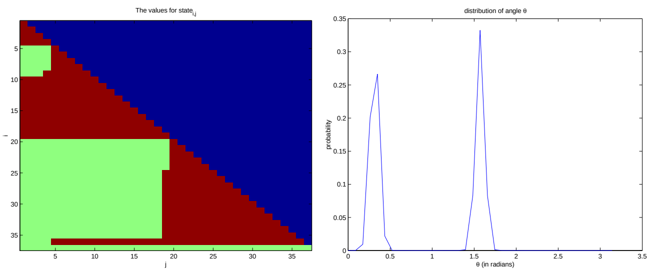

Fig. 4. **LEFT:**The optimal policy for MDP2. The sampling value δ used was 5 degrees, therefore the location i, j in the image above represents the action when the robot is at $i\delta$ and the last reconstruction it made was at $j\delta$. The vertical column is i and the horizontal columns is j. The blue upper right half illustrates the inaccessible states. The green values at the lower right correspond to RECONSTRUCT actions and the red region in between correspond to MOVE action. The distribution P^{θ} is according to the distribution on the **RIGHT**.

The drawback of this approach is the increase in the number of states, and hence the running time of the algorithm. The former policy based on MDP1 requires $2n$ states, whereas the number of states for MDP2 is $\frac{n(n+1)}{2}$. Note that states s_{ij} with $i < j$ are not well defined. The power of MDP2 is illustrated in figure 4, where the figure on the left illustrates the optimal policy for the bimodal distribution P^{θ} on the right. Based on MDP1, in contrast, the robot moves until enough probability accumulates and reconstructs afterwards and does not exploit the low probability region as MDP2. We further illustrate the optimal policies for MDP1 and MDP2 for various distributions in the simulations section.

IV. Simulation Results

In this section we compare the four algorithms we describe in this paper. **UNIREC** and **EXPREC** are the two competitive algorithms described in table I. We will refer to the optimal policy of MDP1 summarized in figure 2 as **POLICY1** and the optimal policy of MDP2 summarized in figure 3 as **POLICY2**.

A. The underlying distribution is known

The algorithms **UNIREC** and **EXPREC** have performance guarantees regardless of the distribution P^{θ}. In this section, we try to answer the question: Is it really worth solving for optimal policies, even when P^{θ} is available? The answer turns out to be yes, as the following experiments show.

We compare the results for MDPs built using the exact distribution of θ with the competitive algorithms. In other words, the instances of the simulations were generated from the distributions in figure 5 and same distributions were used to build the MDPs. The sampling parameter for all the MDPs we used is 5 degrees which is equal to the bucket sizes of the distributions.

In the following experiment, summarized in table II, the robot stands on the wall, 10m away from the corner. This aligns the line of sight of robot with the visible edge, allowing us to use the full range of $[0, \pi]$ for θ. Hence, $D = 10m$. Each reconstruction takes 2 seconds and the robot moves with a speed of $0.5m/s$. The time it takes to reach the corner is 20 seconds, therefore $\delta = 0.1$ for algorithms UNIREC and EXPREC.

P^{θ}	**UNIREC**	**EXPREC**	**POLICY1**	**POLICY2**
1	43.98	39.40	26.72	28.93
2	57.22	39.42	41.77	35.16
3	48.17	34.88	48.53	34.82
4	43.06	37.63	37.69	27.39

TABLE II

THE RESULTS WHEN THE UNDERLYING DISTRIBUTIONS MATCH THE BELIEFS ABOUT THE DISTRIBUTION. 1000 SAMPLES WERE DRAWN FROM THE DISTRIBUTIONS IN FIGURE 5 (COLUMN1). REST OF THE COLUMNS PRESENT THE AVERAGE TIME TO SEE THE NEXT EDGE FOR THE FOUR ALGORITHMS PRESENTED IN THIS PAPER.

Note that Distribution 3, which is uniform in $[0, \pi]$, represents the case when there is no apriori information about the environment. The policies for this case are presented in figure 6. In this case, all MDP1 can do is

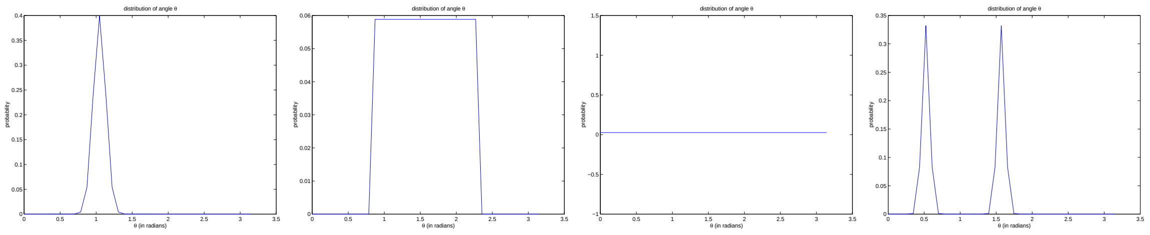

Fig. 5. The distributions used for experiments: Each bucket represents 5 degrees. **Left:** Distribution 1 is generated using a gaussian with mean 60 degrees and variance 5 degrees. **Middle Left:** Distribution 2 is uniform between $\frac{\pi}{4}$ and $\frac{3\pi}{4}$. **Middle Right:** Distribution 3 is uniform between 0 and π. **Right:** A bimodal distribution obtained by adding up two gaussians with means $\frac{\pi}{6}$ and $\frac{\pi}{2}$ and a variance of 3 degrees.

to move until enough probability is accumulated and to reconstruct at every step afterwards, as it has no memory of the previous reconstruction. MDP2, in contrast, prefers to move further after a recent reconstruction.

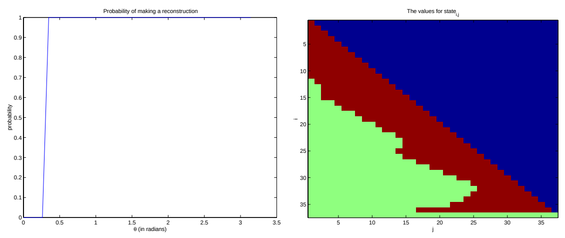

Fig. 6. Optimal policies for MDP1 and MDP2 for Distribution 3 in figure 5. **Left:** The probability of making a reconstruction for MDP1 **Right:** The policy for MDP2

The next experiment is the same as the previous one other than the reconstruction time $\tau = 10$ seconds and therefore $\delta = 0.5$ and the results are presented in table III.

P^θ	UNIREC	EXPREC	POLICY1	POLICY2
1	53.96	40.94	36.99	37.43
2	68.64	46.49	47.86	53.63
3	61.30	40.82	76.21	46.95
4	56.92	35.24	68.25	43.16

TABLE III

THE RESULTS WHEN THE UNDERLYING DISTRIBUTIONS MATCH THE BELIEFS ABOUT THE DISTRIBUTION. 1000 SAMPLES WERE DRAWN FROM THE DISTRIBUTIONS IN FIGURE 5 (COLUMN1). REST OF THE COLUMNS PRESENT THE RESULTS FOR RUNNING THE FOUR ALGORITHMS PRESENTED IN THIS PAPER.

Comparing results in table II and table III we see that if the underlying distribution is available, the optimal policies outperform the competitive algorithms. Another observation is that when the reconstruction is costly ($\tau = 2$ vs $\tau = 10$) the number of reconstructions become really significant and POLICY2 outperforms POLICY1. To illustrate this further we ran simulations that keep the distribution constant but vary the reconstruction time and the results are shown in figure 7.

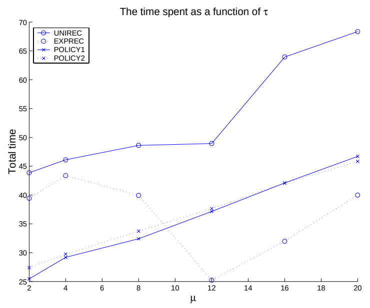

Fig. 7. $P^\theta(\theta) = \mathcal{N}(60, 5)$, but τ varies according to the values in the horizontal axis. POLICY2 outperforms POLICY1 as τ increases.

B. When the beliefs are wrong

In order to illustrate what happens when the robot's beliefs do not match the environment, we use a different distribution to draw samples for the experiment than the one we use to find the optimal policies for the MDPs. For example, for $\mu = 40$ in the left plot of figure 8, we computed the optimal policies for $\mathcal{N}(40, 5)$ and then used 1000 samples from $\mathcal{N}(60, 5)$ for simulations, in order to create a discrepancy between the robot's beliefs and the state of the world.

As in the previous section, in the following experiments the robot stands on the wall, 10m away from the corner. Each reconstruction takes 2 seconds and the speed of the robot is $0.5m/s$. The time it takes to reach the corner is 20 seconds, therefore $\delta = 0.1$ for algorithms UNIREC and EXPREC.

UNIREC	EXPREC	POLICY1	POLICY2
47.44	34.43	38.16	50.87

TABLE IV

ROBOT THINKS THE WORLD IS $\mathcal{N}(60, 5)$ BUT IN FACT THE SAMPLES ARE DRAWN FROM UNIFORMLY FROM $[0, \pi]$.

As expected, when the beliefs are wrong, the performance of the algorithms UNIREC and EXPREC do not get affected, since they do not assume any distribution for the input. However, the results in figure 8 and table IV

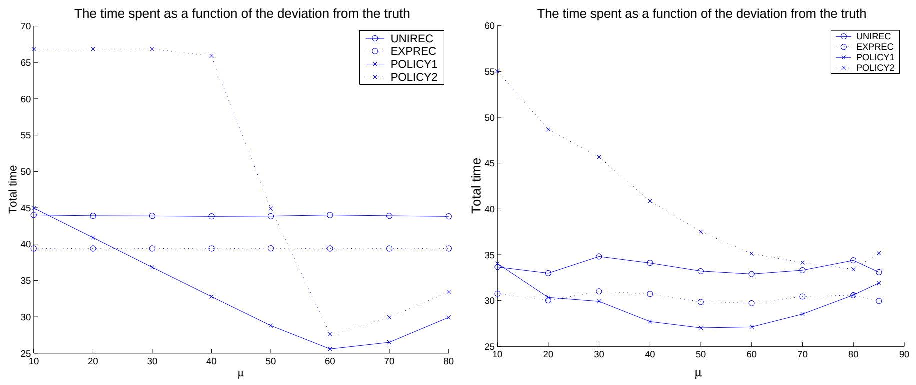

Fig. 8. **LEFT:** The samples representing the real state of the world were drawn from $\mathcal{N}(60, 5)$ and the optimal policies were computed for $\mathcal{N}(\mu, 5)$ where μ is the horizontal axis. Each simulation is the average time to see the next edge for 1000 samples. **RIGHT:** Same as left but the samples representing the real state of the world were drawn uniformly from $[0, \frac{\pi}{2}]$.

suggest that MDP2 is more sensitive to errors in the underlying beliefs than MDP1. This is because MDP2 has a more specialized policy than MDP1.

V. Conclusion

We have studied the problem of how to look around a corner in a polygonal environment given that we want to minimize the time spent in traveling as well as in reconstruction. We addressed local optimality regarding the visibility of the next occluded edge. In this sense, we differ from Best Next View algorithms which guarantee visibility without minimizing the cost of achieving it. Our strategy can accelerate heuristic planning for global exploration.

Our contribution is in the competitive analysis of the problem and its formalization as a Markov Decision Process. In our future work we plan the following thrusts: to incorporate uncertainty in the position estimates of the robot, to relax the circle discretization and search for a more efficient state-action tessellation of the plane, to study the local problem in 3D by generalizing the form of the occluding contour, and finally to formulate global exploration as an MDP.

Acknowledgements: The authors are grateful for support through the following grants: NSF-IIS-0083209, NSF-EIA-0218691, NSF-IIS-0121293, NSF-EIA-9703220, a DARPA/ITO/NGI subcontract to UNC, NSF-CCR98-20885, NSF-CCR01-05337 and a Penn Research Foundation grant. The authors would also like to thank Professor Michael Kearns and Professor Max Mintz for their valuable feedback.

Appendix: The competitive ratio of EXPREC

In this section we derive the competitive ratio for the algorithm EXPREC. Recall that instead of visiting $\delta, 2\delta, 3\delta, \ldots$, the robot takes exponential jumps and visits $\delta, 2\delta, 4\delta, \ldots, 2^i\delta$. During the i^{th} step robot traverses an angle of $\delta 2^{i-1}$ and the total angle traversed so far is $\delta(2^i - 1)$. Therefore if θ is the real angle, the robot sees the next edge as soon as it takes $i = \lceil \log(\frac{\theta}{\delta} + 1) \rceil$ steps.

Case 1: $\theta \leq \frac{\pi}{4}$

$$\begin{aligned}
C &= \frac{i\tau + \delta(2^i - 1)\frac{R}{v}}{\tau + \frac{D}{v}\sin\theta} \\
&= \frac{i\tau + \delta(2^i - 1)\frac{\tau}{\delta}}{\tau + \frac{\tau}{\delta}\sin\theta} \\
&= \frac{i + (2^i - 1)}{1 + \frac{\sin\theta}{\delta}} \\
&= \frac{\lceil \log(\frac{\theta}{\delta} + 1) \rceil + 2^{\lceil \log(\frac{\theta}{\delta}+1) \rceil} - 1}{1 + \frac{\sin\theta}{\delta}} \\
&\leq \frac{(\log(\frac{\theta}{\delta} + 1) + 1) + 2^{(\log(\frac{\theta}{\delta}+1)+1)} - 1}{1 + \frac{\sin\theta}{\delta}} \\
&= \frac{\log(\frac{\theta}{\delta} + 1) + 2^{(\log(\frac{\theta}{\delta}+1)+1)}}{1 + \frac{\sin\theta}{\delta}} \\
&= \frac{\log(\frac{\theta}{\delta} + 1) + 2(\frac{\theta}{\delta} + 1)}{1 + \frac{\sin\theta}{\delta}} \\
&= \frac{2(\frac{\theta}{\delta} + 1)}{1 + \frac{\sin\theta}{\delta}} + \frac{\log(\frac{\theta}{\delta} + 1)}{1 + \frac{\sin\theta}{\delta}} \\
&= \frac{\frac{2}{\delta}(\theta + \delta)}{\frac{1}{\delta}(\delta + \sin\theta)} + \frac{\log(\frac{\theta}{\delta} + 1)}{1 + \frac{\sin\theta}{\delta}} \\
&= \frac{2(\theta + \delta)}{(\delta + \sin\theta)} + \frac{\log(\frac{\theta}{\delta} + 1)}{1 + \frac{\sin\theta}{\delta}}
\end{aligned}$$

Which achieves its maximum value of 2.2214 when $\delta \rightarrow 0$ and $\theta = \frac{\pi}{4}$.

Case 2: $\frac{\pi}{4} \leq \theta \leq \frac{\pi}{2}$

If θ is slightly larger than $\frac{\pi}{4}$, the robot takes a huge last step and goes all the way to the corner following the entire half circle. The competitive ratio is:

$$\begin{aligned} C &\leq \frac{\log(\lceil\frac{\pi}{2\delta}\rceil)\tau + \frac{\pi R}{2v}}{\tau + \frac{D}{v}\sin\frac{\pi}{4}} \\ &= \frac{\log(\lceil\frac{\pi}{2\delta}\rceil)\tau + \frac{\pi\tau}{2\delta}}{\tau + \frac{\tau}{\delta}\sin\frac{\pi}{4}} \\ &= \frac{\log\lceil\frac{\pi}{2\delta}\rceil + \frac{\pi}{2\delta}}{1 + \frac{\sin\frac{\pi}{4}}{\delta}} \\ &= \frac{\log\lceil\frac{\pi}{2\delta}\rceil}{1 + \frac{\sin\frac{\pi}{4}}{\delta}} + \frac{\frac{\pi}{2\delta}}{1 + \frac{\sin\frac{\pi}{4}}{\delta}} \end{aligned}$$

As $\delta \rightarrow 0$, the first term vanishes and the competitive ratio becomes $\frac{\pi/2}{\sin\frac{\pi}{4}} = 2.2214$

VI. REFERENCES

[1] S. Albers and K. Kursawe. Exploring unknown environments with obstacles. In *In Proc. 9th ACM-SIAM Sympos. Discrete Algorithms, 1998. 13*, 1998.

[2] G. J. W. Amos Fiat. *Online algorithms : the state of the art*. Berlin ; New York : Springer, 1998.

[3] D. P. Bertsekas. *Dynamic Programming and Optimal Control: 2nd Edition*. Athena Scientific, 2000.

[4] A. Blum, P. Raghavan, and B. Schieber. Navigating in unfamiliar geometric terrain. *SIAM Journal on Computing*, 26(1):110–137, 1997.

[5] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press New York, NY, USA, 1998.

[6] W. Burgard, D. Fox, M. Moors, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Proc. of International Conference on Robotics and Automation*, San Fransisco, 2000.

[7] H. Choset and J. Burdick. Sensor-based exploration: The hierarchical generalized voronoi graph. *The International Journal of Robotics Research*, 19(2):96–125, 2000.

[8] X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment i: The rectilinear case. *Journal of the ACM*, 45:215–245, 1998.

[9] X. Deng and A. Mirzaian. Competitive robot mapping with homogeneous markers. *IEEE Trans. on Robotics and Automation*, 12(4):532–542, 1996.

[10] H. Gonzalez-Banos, A. Efrat, J. C. Latombe, E. Mao, and T. M. Murali. Planning robot motion strategies for efficient model construction. In *Proc. 9th International Symposium of Robotics Research*, Victoria, Australia, 2001.

[11] F. Hoffmann, C. Icking, R. Klein, and K. Kriegel. The polygon exploration problem. *SIAM Journal on Computing*, 31(2):577–600, 2002.

[12] C. Icking, R. Klein, and L. Ma. How to look around a corner. *Proc. of the 5th Canadian Information Processing Society Congress*, pages 443–448, 1993.

[13] S. Koenig, C. Tovey, and W. Halliburton. Greedy mapping of terrain. In *Proc. of International Conference on Robotics and Automation*, pages 3594–3599, 2001.

[14] I. M. Rekleitis, G. Dudek, and E. E. Milios. Multi-robot collaboration for robust exploration. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):7–40, 2001.

[15] I. Stamos and P. K.Allen. Integration of range and image sensing for photorealistic 3d modeling. In *Proc. of International Conference on Robotics and Automation*, pages 1435–1440, San Fransisco, 2000.

[16] R. S. Sutton and A. G. Barto. *Reinforcement Learning:An Introduction*. The MIT Press, 1998.

[17] C. Taylor and D. Kriegman. Vision-based motion planning and exploration algorithms for mobile robots. *IEEE Trans. On Robotics and Automation*, 14(3):147–427, 1998.

[18] S. Thrun. Robotic mapping: A survey. Technical Report CMU-CS-02-111, Dept. of Comp. Science, Carnigie Melon University, 2002.

[19] R. M. Zlot, A. T. Stentz, M. B. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. In *IEEE International Conference on Robotics and Automation*, May 2002.