Planning Expected-time Optimal Paths for Searching Known Environments

Alejandro Sarmiento Rafael Murrieta-Cid Seth Hutchinson
Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign
Urbana Illinois, USA
Email: {asarmien, murrieta, seth}@uiuc.edu

***Abstract*— In this paper we address the problem of finding time optimal search paths in known environments. In particular, the task is to search a known environment for an object whose unknown location is characterized by a known probability density function (pdf). With this formulation, the time required to find the object is a random variable induced by the choice of search path together with the pdf for the object's location. The optimization problem is to find the path that yields the minimum expected value of the time required to find the object.**

We propose a two layered approach. Our algorithm first determines an efficient ordering of visiting regions in a decomposition that is defined by critical curves that are related to the aspect graph of the space to be searched. It then generates locally optimal trajectories within each of these regions to construct a complete continuous path. We have implemented this algorithm and present results.

I. Introduction

In this paper, we address the problem of finding an object in a polygonal environment as quickly as possible on average with a mobile robot that can sense the environment continuously. 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.

In [10], we presented a discrete, combinatoric version of this problem. In that work, we abstracted the problem to one of finding a path in a graph whose nodes represented sensing locations (guards). Associated to each node is the probability of sensing the object from the corresponding location, and associated to each arc is a cost that corresponds to the time required to move between the corresponding sensing locations. We showed that for this problem, a trajectory that minimizes the distance traveled may not minimize the expected value of the time to find the object.

In [11], we extended our approach to the more general case of searching in a polygon. In this case, we used a visibility-based decomposition of the polygon to again convert the problem into a combinatoric one. The visibility regions were used to calculate the probability of seeing an object for the first time from particular sensing locations, which were again chosen from a predefined set. Paths were constructed from arcs in a reduced visibility graph. We showed the problem to be NP-hard by reduction.

The work that we present here is qualitatively different from our previous efforts. In our past work, all robot paths consisted of piecewise linear segments between preselected sensor locations. In this paper, we find *optimal continuous paths* by using methods derived from the calculus of variations, and there is no predefined set of sensing locations for the given task. Our approach exploits a decomposition of the space to be searched using critical curves that are related to the aspect graph of the space. We use the utility function described in [11] to find an efficient order of visiting these regions, and the calculus of variations to find locally optimal trajectories within each one of them.

II. Problem Definition

As in [11], we define the the random variable T to be the time required to find the object, i.e., the time until the object enters the robot's field of view for the first time. We are interested in finding a continuous path S that minimizes the expected value of this random variable along that path, $E[T|S]$. This trajectory will, on average, find the object as quickly as possible.

We model the robot as a single point with an infinite range, omni-directional sensor. We do not impose any constraints on the movement of the robot other than constant speed.

III. Proposed Approach for Continuous Sensing

As mentioned before, in this paper we are dealing with *continuous sensing* in a continuous space. We assume that the robot is sensing the environment as it moves. This contrasts with sensing only at specific locations, as was described in [11].

We say that a continuous trajectory *covers* [12] a polygon P if each point $p \in P$ is visible from some point along the trajectory. If the trajectory is to minimize the distance traveled, then the problem is called the Shortest Watchman Tour problem [1]. This is not exactly our problem since, as we showed in [10], a trajectory that minimizes the distance traveled may not minimize the expected value of the time to find an object along it.

The Shortest Watchman Tour problem is also related to the Art Gallery problem [5] in that they both deal with visibility in polygons. However, the Art Gallery problem seeks to minimize the number of point guards needed cover a polygon and is not concerned at all with the distances between them.

Any trajectory that covers a simple (without holes) polygon must visit each subset of the polygon that is bounded by the aspect graph lines associated to non-convex vertices of the polygon. An aspect graph for a polygon [3] consists of a set of line segments generated by features of the polygon. We only use the line segments generated by non-convex vertices of the polygon. These line segments are simply extensions of the incident edges on non-convex vertices, as shown in Fig. 1 with grey lines.

We call the area bounded by these aspect graph lines the *corner guard regions*. These regions have the characteristic that any point inside them can see "both sides" of their associated non-convex vertices. Therefore, a continuous trajectory that covers a simple polygon needs to have at least one point inside the region associated to "outlying" non-convex vertices (non-convex vertices in polygon ears), like A and C in the figure. Since these points need to be connected with a continuous path, the trajectory will cross all other corner guard regions, like the one associated to vertex B.

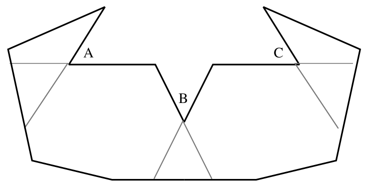

Fig. 1. The aspect graph lines (in grey) associated to non-convex vertices A, B and C

Since a continuous trajectory needs to visit all the corner guard regions, it is important to decide in which order they are visited. The problem can be abstracted to finding an specific order of visiting nodes in a graph that minimizes the expected value of time to find an object. In [11] we showed that a version of this problem is NP-hard.

For this reason, to generate continuous trajectories we propose an approach with two layers that solve specific parts of the problem. The high level, *combinatoric* layer attempts to find a "suitable" order of visiting corner guard regions without taking into account how exactly the robot is to move between them. The low level, *continuous* layer takes an ordering from the upper level and tries to find how to best visit the given regions.

This decoupling makes the problem simpler to address, but does so at the expense of global optimality. To preserve global optimality, an algorithm would need to consider how the robot is moving while generating the best ordering of visiting corner guard regions. Calculating the globally optimal robot motions is not an easy task because the trajectory can make "sudden" direction changes within a single region (as will be described in section V) and also because these motions must make a compromise between the distance traveled (time) and the amount of the environment (probability) that is visible at different points along the complete trajectory (to minimize the overall expected value).

In the remainder of the paper, we describe our two-level approach. In section IV we describe how optimal continuous paths are generated for moving from one region to another. In section V we address the problem of choosing a good ordering for the regions. Results are given in section VI.

IV. Planning Optimal Continuous Paths within Regions

Once an ordering of visiting corner guard regions is established by the combinatoric layer, it is still necessary to generate a continuous trajectory between them.

Given that we want to generate a continuous path, it is necessary to compute the expected value of time $E[T|S]$ along a trajectory S (as the robot moves). The form of the equation to compute $E[T|S]$ changes in different regions of the polygon. For this reason, we have analyzed the simplest case – within one region – and concatenated these sections for a complete trajectory. Note that with this approach, there are no guarantees as to whether locally optimal sub-paths will lead to a globally optimal solution.

A. Continuous Sensing in the Base Case

The simplest case for a continuous sensing robot is that shown in Fig. 2. In this case, the robot has to move around a non-convex vertex (corner) to explore the unseen area A'. For now, we assume that this is the only unseen portion of the environment.

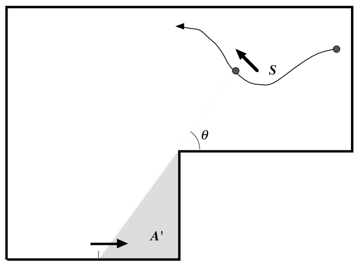

Fig. 2. Base case for a continuous sensing robot

As the robot follows any given trajectory S, it will sense new portions of the environment. The rate at which new environment is seen determines the expected value of the time required to find the object along that route. In particular, consider the following definition of expectation for a non-negative random variable from [8],

$$E[T|S] = \int_0^\infty P(T > t)\ dt. \tag{1}$$

The particular route S followed by the robot determines the probability of not having seen the object at any given time, $P(T > t)$.

We require that this probability decreases monotonically, which is equivalent to considering only trajectories along which the size of the unseen region decreases monotonically. As shown in Fig. 2, the remaining section of the environment to be explored A' decreases monotonically if and only if the angle from the corner to the robot increases monotonically. For this reason, it is natural to express the trajectory in polar coordinates with the origin at the corner.

B. Expected Value of Time Along any Trajectory

In the simple environment shown in Fig. 3 the robot's trajectory is expressed as a function in polar coordinates with the origin on the non-convex vertex. We assume that the robot will have a starting position such that its line of sight will only sweep the horizontal edge E_1. As mentioned before, the expected value of the time to find an object depends on the area A' not yet seen by the robot. Since we assume infinite sensing range, at any time t the only important feature of the robot's position is its angle $\theta(t)$ relative to the origin.

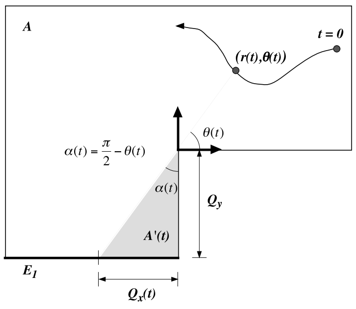

Fig. 3. Base case for a continuous sensing robot

The following analysis is only valid for an axis-parallel edge E_1, but it can be easily adapted to the general case. Let $Q_x(t)$ and Q_y be horizontal and vertical distances from the origin to the point where the robot's line of sight through the origin intersects E_1. The area of the unexplored region $A'(t)$ is

$$A'(t) = \frac{Q_y \; Q_x(t)}{2}. \tag{2}$$

As can be seen in Fig. 3,

$$\tan(\alpha(t)) = \frac{Q_x(t)}{Q_y}$$

and

$$\alpha(t) = \frac{\pi}{2} - \theta(t).$$

Since $\tan\left(\frac{\pi}{2} - \theta(t)\right) = \frac{1}{\tan(\theta(t))}$, we have $\tan(\theta(t)) = \frac{Q_y}{Q_x(t)}$, and (2) can be written as

$$A'(t) = \frac{Q_y \; Q_x(t)}{2} = \frac{{Q_y}^2}{2 \; \tan(\theta(t))}.$$

Assuming that the probability density function of the object's location over the environment is constant, the probability of not having seen the object at time t is

$$P(T > t) = \frac{A'(t)}{A} = \frac{{Q_y}^2}{2A \; \tan(\theta(t))}, \tag{3}$$

where A is the area of the whole environment.

Finally, from (1) and (3),

$$E[T|S] = \frac{{Q_y}^2}{2A} \int_0^{t_f} \frac{dt}{\tan(\theta(t))}. \tag{4}$$

Equation (4) is useful for calculating the expected value of the time to find an object given a robot trajectory S expressed as a parametric function $\theta(t)$. It is interesting to note that the expression does not directly depend on the radius $r(t)$ (a consequence of infinite sensing range). In the next section, we will also use (4) to find the optimal trajectory S^* by minimizing the value of the integral.

C. Minimization Using Calculus of Variations

The calculus of variations is a mathematical tool employed to find stationary values (usually a minimum or a maximum) of integrals of the form

$$I = \int_a^b F(x, y, y') \; dx, \tag{5}$$

where x and y are the independent and dependent variables respectively.

The integral in (5) has a stationary value if and only if the Euler-Lagrange equation is satisfied,

$$\frac{\partial F}{\partial y} - \frac{d}{dx}\left(\frac{\partial F}{\partial y'}\right) = 0. \tag{6}$$

In our case, it is not useful to apply the prototypical Euler-Lagrange equation directly to expression (4) for two reasons. First, r and θ are expressed as parametric equations, instead of one as a function of the other. This is not really a problem, because expressions very similar to (6) can be derived to accommodate the case of parametric functions [2]. The real problem is that (4) does not impose any constraints on the parametric equations describing the robot motion. The optimal trajectory without any constraints would be one where θ increases infinitely fast.

To address both of these problems, we introduce the constraint that the robot moves with constant (unitary) speed. To do this, we express its velocity vector as a generalized motion [7] in a basis where one component U_r is radial from the origin and the other U_θ is perpendicular, as shown in Fig. 4. Both U_r and U_θ are unit vectors that define an orthogonal basis. In this basis, the robot's velocity (in polar coordinates) can be described as

$$V = \dot{r} \; U_r + r \; \dot{\theta} \; U_\theta.$$

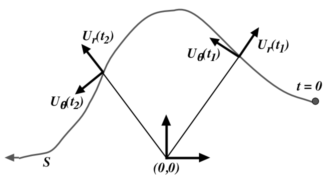

Fig. 4. Generalized motion of a particle moving along path S

The constraint that the robot speed is constant can be expressed as

$$\|V\| = \dot{r}^2 + r^2\ \dot{\theta}^2 = 1. \tag{7}$$

In practice, this means that the maximal speed the robot can achieve is constant regardless of the direction of motion. The velocity components need to "add up" to a constant value. This contrasts with other systems where each dimension can be controlled independently, like a plotter.

Starting with equation (7), it is possible to express the differential of time as a function of a differential of θ. This will allow us to rewrite the parametric equation as a function in which θ and r are the independent and dependent variables respectively,

$$\begin{aligned} 1 &= \frac{(dr)^2}{(dt)^2} + r^2\ \frac{(d\theta)^2}{(dt)^2}, \\ (dt)^2 &= \left((dr)^2 + r^2\ (d\theta)^2\right)\ \frac{(d\theta)^2}{(d\theta)^2}, \\ (dt)^2 &= \left(r'^2 + r^2\right)\ (d\theta)^2, \\ dt &= \left(r'^2 + r^2\right)^{\frac{1}{2}}\ d\theta, \end{aligned} \tag{8}$$

where $r' = \frac{dr}{d\theta}$. Substituting (8) into (4), we obtain an expression for the expected value of time to find an object where the robot's trajectory S is expressed as r being a function of θ,

$$E[T|S] = \frac{{Q_y}^2}{2A} \int_{\theta_i}^{\theta_f} \frac{1}{\tan(\theta)} \left(r'^2 + r^2\right)^{\frac{1}{2}}\ d\theta. \tag{9}$$

To find stationary values of (9), we use (6) with $x = \theta$, $y = r$ and $F = \frac{1}{\tan\theta}\left(r'^2 + r^2\right)^{\frac{1}{2}}$. After simplification, this yields the following second order non-linear differential equation,

$$r'' = r + \frac{2r'^2}{r} + \frac{2}{\sin(2\theta)}\left(r' + \frac{r'^3}{r^2}\right). \tag{10}$$

This equation describes the route to move around a non-convex vertex (corner) to search the area on the other side optimally (according to the expected value of time).

D. Numerical Integration

We solved equation (10) numerically using an adaptive step-size Runge-Kutta method [6]. Since this equation is of second order, any numeric approach that integrates it as an initial value problem requires two initial conditions: $r(\theta_i)$ and $r'(\theta_i)$. We know the staring point $r(\theta_i)$ and the integration range (θ_i, θ_f), but we do not impose any other constraints on the trajectories other than unitary speed. Therefore, the possible solutions are a family of curves that depends on the value of the first derivative at the beginning of the integration range $r'(\theta_i)$. These are shown in Fig. 5.

Most of the possible solutions diverge long before they reach the end of the integration range. In fact, it is evident from (10) that the solution is not defined there (at $\theta_f = \frac{\pi}{2}$). However, it is possible to get arbitrarily close, and to do so, the first derivative at the end of the integration range must be such that the trajectory approaches the target manifold (the vertical line in Fig. 3) perpendicularly. This translates to stating that $r'(\theta_f) = 0$. In fact, the transversality condition for the Euler-Lagrange equation establishes that, in order to satisfy the equation and obtain a minimum, the solution function must be perpendicular to the target manifold at $t = t_f$ [9].

This observation allows us to integrate equation (10) as a two point boundary value problem, where we specify the position at the beginning of the integration range $r(\theta_i)$ and the first derivative at the end $r'(\theta_f)$. For this, we coupled the Runge-Kutta algorithm with a globally convergent Newton-Raphson method [6].

Fig. 6 shows the trajectories generated for six different starting positions in solid black lines. To save space, the figure only shows the upper right portion of an environment similar to that in Fig. 3 (the units on the axes are arbitrary).

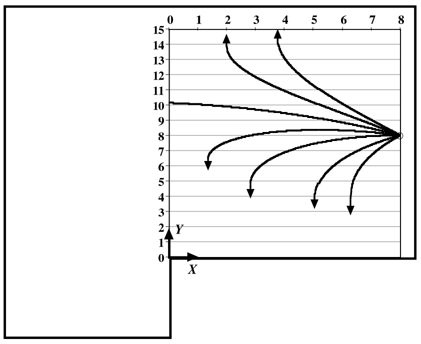

Fig. 5. Family of curves depending on initial conditions

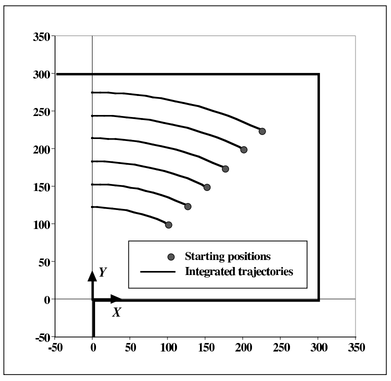

Fig. 6. Optimal trajectories for a simple environment obtained through numerical integration

E. Simulated Annealing

To corroborate our results from the previous section, we found an approximate solution to the original problem, as depicted in Fig. 2, by another method independent of our previous analysis. For this, we implemented Simulated Annealing [4]. This stochastic relaxation method is based on describing the possible solutions as system states, assigning an energy value to them and then minimizing that energy. Obviously, the state of minimum energy must correspond to the optimal solution.

The system goes through a heating process, where the energy of the system is incremented then a cooling process where the energy is iteratively decreased. We made these variations exponentially by multiplying the current temperature by constants (2.0 for heating up, 0.98 for cooling down).

To generate a new state s_{i+1}, the current state s_i is perturbed randomly. If the energy of the new state $U(s_{i+1})$ decreases, then the new state is deterministically accepted. If the energy rises, a Boltzmann acceptance criterion is used. Namely, the new state is accepted with probability

$$P \sim e^{K\left(\frac{U(s_i)-U(s_{i+1})}{Temp}\right)},$$

where K is a normalization constant.

In our problem, a state is an ordered set of control points that define the robot's trajectory. These points were randomly perturbed along one dimension only (vertically) – except for the first point, which corresponds to the initial position, and was always fixed. We defined the energy of any state as the expected value of time to find an object by visiting those points in order (with respect to the horizontal dimension).

We started the heating process with unitary temperature (in arbitrary units) and multiplied it by 2.0 until more than 97% of the new states generated were accepted. Then, we cooled down with a constant of 0.98 until no new states were accepted. Each epoch corresponded to individually perturbing each point approximately four times, and then, at the end of the epoch, perturbing the whole range once.

As can be seen in Fig. 7, the general shape of the trajectories generated for six different starting positions by our numerical integration (solid lines) and stochastic relaxation (control points) are very similar. We should point out, however, that each Simulated Annealing run took more than an hour whereas the numeric integration is done in a fraction of a second. As mentioned before, the figure only shows the upper right section of an environment, like that in Fig. 3 of arbitrary dimensions.

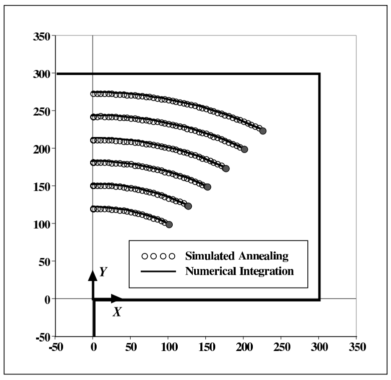

Fig. 7. Optimal trajectories for a simple environment obtained with simulated annealing

V. Choosing an Ordering of Regions

To cover a simple polygon, it is sufficient that a trajectory visits at least one point inside each corner guard region (as defined in section III) associated to reflex vertices of the polygon. The high level, combinatoric layer attempts to find an ordering for the robot to visit these corner guard regions such that the expected value of the time to find an object in the environment is reduced.

To find a suitable ordering, we defined a point guard inside each corner guard region and used the approach we presented in [11] for sensing at specific locations. Potentially, any point in the closure of a corner guard region can be used as a point guard; we decided to place them very close to the non-convex vertices.

Once point guards have been defined, it is straightforward to calculate the visibility regions and distances required for the proposed utility function of [11]. In the end, the algorithm yields an ordering for visiting corner guard regions (associated to non-convex vertices) that attempts to reduce the expected value of the time to find an object.

Once an ordering has been established, the lower level, continuous layer uses the sequence of non-convex vertices to perform locally optimal motions around each of them, thus generating a complete trajectory that covers the polygonal environment.

We believe that this trajectory reduces the expected value of the time to find an object, but we know that any trajectory generated in this fashion will not be globally optimal in the general case. There are several reasons for this, the most obvious being that any partition of the problem into locally optimal portions does not guarantee global optimality (Bellman's principle of optimality does not apply). Another reason is that our generated trajectories will only change direction abruptly (non-smoothly) in aspect graph lines and points where the edge being seen through a reflex vertex changes, for example, when the robot has finished seeing an ear of the polygon and must reverse direction. However, an optimal trajectory might need to change direction in other points of the environment as well, as described in the next section.

A. Direction Changes within a Region

An optimal trajectory may change direction at points that are not part of our set of critical curves, that is, it may change direction abruptly in the interior of a region. Consider the polygon in Fig. 8(a). If the robot starts at P, it is clear that any trajectory that covers the polygon must reach the vertical lines at non-convex vertices A and B belonging to the aspect graph. If such a trajectory is to be optimal, then it is not necessary to cross said lines because the complete side region would already be visible and nothing is to be gained by going further.

Also, since both non-convex vertices are "above" the starting position and the robot is already at the lower limit of the polygon, the optimal trajectory will not move vertically. The robot cannot go "downward" because it would leave the polygon, and it is not useful to move "upward" because no new regions would ever be visible moving this

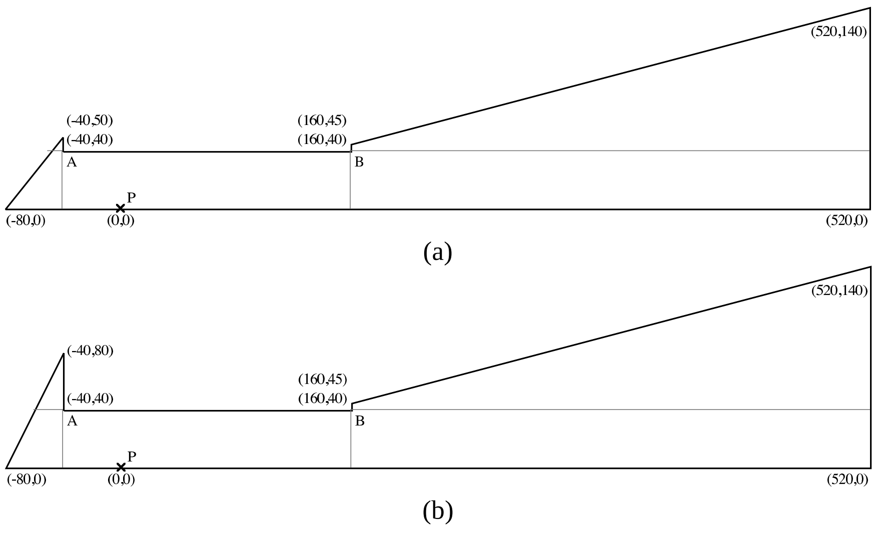

Fig. 8. Two similar polygonal environments

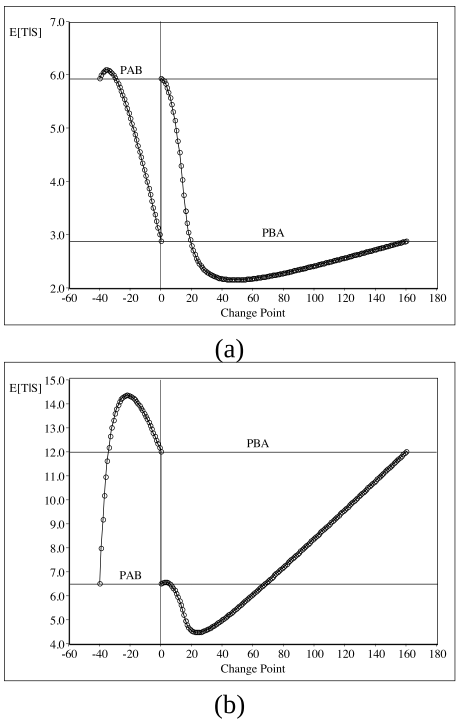

Fig. 9. Expected value of time vs. direction change point

way. Any new point seen in a diagonal trajectory could be seen sooner by traveling a shorter distance on the horizontal component of said trajectory. Also, the two closest aspect graph lines that must be visited are vertical, therefore, moving vertically does not decrease the distance to either of them.

Since moving vertically does not help either of the two variables involved in the expected value search, distance (time) and visible area (probability), we can conclude that the optimal trajectory will not have a vertical component.

Having established that the optimal trajectory for this particular problem will only move horizontally, let us consider the case of horizontal trajectories starting at P that cover the whole polygon.

A trajectory that covers the polygon and only moves horizontally needs to reverse direction *at least* once. If it only changes direction once, then there are only 2 cases, it must go from P to B then to A (PBA) or go from P to A then to B (PAB).

It is also possible that a trajectory changes direction twice, for example, it may go from P towards (but not reaching) B, reverse once to go to A then reverse again to reach B. In this case, there is a range of possible trajectories depending on how far they go the first time they move towards B.

The graph in Fig. 9(a) tallies the expected value of the time to find an object in the environment depicted in Fig. 8(a) when following a horizontal trajectory that makes only two direction reversals. The horizontal axis in Fig. 9(a) represents the point at which the first direction change is made. The negative domain means that the robot starts moving towards A instead of B.

The discontinuity at the origin is the result of the initial direction of motion. Since the trajectories must make exactly two direction changes, if the robot initially moves right towards B, it will reverse direction and reach A first. On the other hand, if it starts moving left for a while, it will change direction and reach B first.

Notice that if the first direction change happens very close to the origin or very close to A or B, the expected value is practically the same as either of the two trajectories with only one change (shown in the graph as the horizontal lines). It is also interesting to note that the best case (lowest expected value) for these trajectories does not happen at the boundary – equivalent to the PBA and PAB trajectories. The optimal trajectory, therefore, must have *at least* two direction changes (it may have more).

Now consider the polygon in Fig. 8(b). It is very similar to (a), except for one vertex that was raised so that the relative area of the left side of the polygon is larger. Analogous to the previous case, Fig. 9(b) shows the expected value of the time to find an object following trajectories with two direction changes in this new polygon. It can be seen that trajectory PAB is now better than trajectory PBA but neither of them is the best one in this group. Also, it is evident that the best point to make the direction change (global minimum) has shifted to the left.

The graphs in Fig. 9 show that, for routes that make two direction changes, the best trajectory (the one with the lowest expected value of time) moves towards but not reaches B, then reverses direction to go to A and then finally B. The best point at which the first direction change is made does not correspond to any point in the aspect graph of the polygon. Furthermore, this point shifted to the left when the area of the left portion of the environment was increased.

These examples do not show what the optimal trajectory is, however, they do show that it must have at least two direction changes and that the points at which these changes are made do not necessarily correspond to aspect graph lines or points where the edge being seen through a reflex vertex changes. In conclusion, an optimal trajectory may change direction abruptly inside our defined regions, not just at the boundary on our defined critical curves.

VI. Simulation Results

This section presents an example of how our proposed two layered approach can be used to generate a continuous trajectory that covers a simple polygon with the goal of reducing the expected value of the time to find an object along that trajectory.

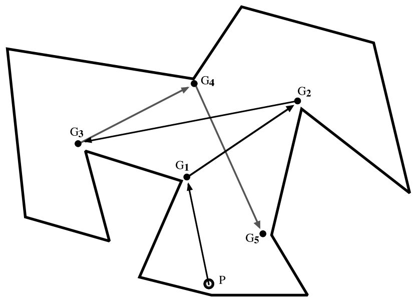

Fig. 10. A simple polygon with non-convex vertices as guards

Fig. 10 shows a simple polygon and a staring position P (near the bottom). We placed a guard G_i close to every non-convex vertex and used the algorithm proposed in [11] to find an efficient ordering for visiting the guard locations. This algorithm returns a complete ordering (all guards are included once).

However, the guard set can be redundant and since sensing is done continuously the polygon may be completely covered before all guards are "visited". In consequence, some guards late in the ordering may not need to be visited. This is the case of guards G_4 and G_5 in the figure.

Once an ordering has been established, the trajectory is generated piecewise according to which guard is to be visited. The robot does not actually travel towards the guard, but rather it goes around its associated non-convex vertex in a locally optimal trajectory, as described in section IV. A locally optimal portion of the complete path is generated for every edge seen through the current non-convex vertex. For example, in Fig. 11 as the robot moves from the starting position P, in the shaded region, the section of the environment that will be visible through guard G_1 is bounded by edge E_1, that is, as the robot moves, its line of sight through the corner will "sweep" edge $E1$ until it reaches edge E_2. At this point, the shape of the current sub-path changes as it is now edge E_2 that will be swept. When the trajectory reaches one of the aspect graph lines associated to the non-convex vertex of the current guard, the process starts over with the next guard in the ordering.

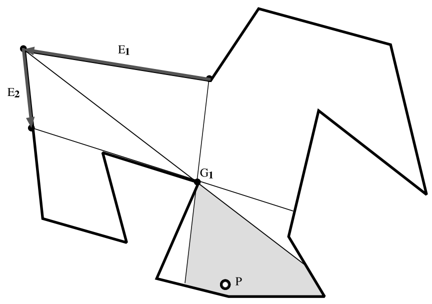

Fig. 11. Edges visible through guard G_1

Fig. 12 shows all the trajectory pieces (A through F) generated for the polygon and the guard they correspond to. There may be occasions, such as portion E, where the locally optimal path would leave the polygon. In this case, the trajectory is saturated and made to follow the polygon boundary. Note that the endpoints of each trajectory portion correspond to critical events, which occur at aspect graph lines or when there is a transition between the edges that are currently been seen through the corner (guard).

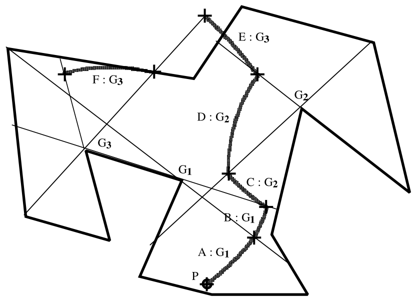

Fig. 12. Locally optimal trajectories for the guards that generated them

Fig. 13 shows the final trajectory for that polygon. It is important to remark that this trajectory attempts to minimize the expected value of the time to find an object, *not* the distance traveled. The zig-zag motion is not necessarily bad because a good trajectory must find a compromise between advancing to the next guard and sensing a larger portion of the environment as soon as possible.

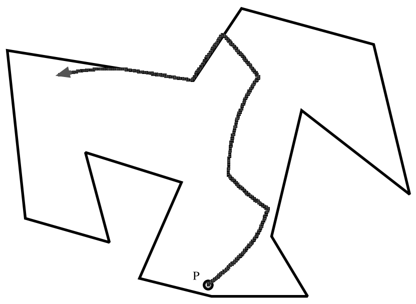

Fig. 13. The final trajectory for a simple polygon

For this particular example, the expected value of the time along the shown trajectory is 115.3. This contrasts with the expected value along the straight line segments shown in Fig. 10 ($G_1 \rightarrow G_2 \rightarrow G_3$), which turns out to be 136.9.

VII. Discussion and Conclusions

We addressed the problem of continuous sensing for expected value search in simple polygons. This problem involves the generation of a motion strategy that minimizes the expected value of the time to find an object.

We presented a two layered algorithm that determines an efficient ordering of visiting regions and then generates locally optimal sub-paths to construct a complete trajectory.

The final trajectory is not globally optimal for two reasons. First, the discrete version of the problem is NP-hard and we proposed a tractable algorithm. Second, we chose to decouple the task of finding an ordering and moving between regions (bounded by aspect graph lines and points where the edge being seen through a reflex vertex changes). However, these lines are not the only places where an optimal trajectory may change direction, showing our proposed decoupling may not be the best one.

Obviously, the optimal paths will depend on the general shape of the polygon. For example, in polygons where most of the area is visible towards the end of the trajectory a motion strategy that moves the robot in the visibility graph will yield good results. This happens because it is reducing the distance to travel up to the point where it is more likely to find the object. In contrast, if the majority of the visible area lies near the starting point a *completely* greedy algorithm that follows the visibility gradient will perform better. In our case, the high level, combinatoric layer attempts to find global optimality by forcing a specific ordering for the low level, continuous layer. Without this ordering, the end result would be a purely greedy algorithm that does not consider the amount of area visible in the future. For this reason, we think our algorithm presents a good trade-off.

References

[1] Chin, W.P. and S. Ntafos, "Optimum Watchman Routes," *Information Processing Letters,* Vol. 28, pp. 39–44, 1988.

[2] Fox, C., *An Introduction to the Calculus of Variations,* Dover Publications, Inc, 1987.

[3] Gigus, Z. and H. Malik, "Computing the Aspect Graph for Line Drawings of Polyhedral Objects" *in Proc. IEEE Conf. on Computer Vision and Pattern Recognition 1998.*

[4] Kirkpatrick, S., C.D. Gelatt Jr and M.P. Vecchi, "Optimization by Simulated Annealing," *Science,* Vol. 220, No. 4598, 1983.

[5] O'Rourke, J., *Art Gallery Theorems and Algorithms,* Oxford University Press, 1987.

[6] Press, W.H. *et al*, *Numerical Recipes in C : The Art of Scientific Computing,* Cambridge University Press, 1993.

[7] Resnik, R. and D. Halliday, *Physics,* John Wiley and Sons, Inc, 1977.

[8] Ross, S.M., *Introduction to Probability and Statistics for Engineers and Scientists,* Wiley, 1987.

[9] Sage, A.P. and C.C. White, *Optimum Systems Control,* Prentice Hall, 1977.

[10] 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.*

[11] Sarmiento, A., R. Murrieta and S.A. Hutchinson, "An Efficient Strategy for Rapidly Finding an Object in a Polygonal World," *in Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems 2003.*

[12] Shermer, T.C., "Recent Results in Art Galleries," *Proc. of the IEEE,* Vol. 80, issue 9, September 1992.