Finding an Unpredictable Target in a Workspace with Obstacles

Steven M. LaValle David Lin Leonidas J. Guibas Jean-Claude Latombe Rajeev Motwani

Computer Science Department
Stanford University
Stanford, CA 94305
{lavalle,dlin,guibas,latombe,motwani}@cs.stanford.edu

Abstract

This paper introduces a visibility-based motion planning problem in which the task is to coordinate the motions of one or more robots that have omnidirectional vision sensors, to eventually "see" a target that is unpredictable, has unknown initial position, and is capable of moving arbitrarily fast. A visibility region is associated with each robot, and the goal is to guarantee that the target will ultimately lie in at least one visibility region. Both a formal characterization of the general problem and several interesting problem instances are presented. A complete algorithm for computing the motion strategy of the robots is also presented, and is based on searching a finite cell complex that is constructed on the basis of critical information changes. A few computed solution strategies are shown. Several bounds on the minimum number of needed robots are also discussed.

1 Introduction

Have you ever searched for someone in a building, possibly exploring the same places multiple times, while finally fearing that the person might have moved to a location already explored? Have you wondered how many searchers it would take to be able to guarantee that the person will eventually be located? This paper presents a formal study of problems of this type, for which the task is to plan a motion strategy that will ensure that a target will eventually be "seen" in a workspace that is cluttered with obstacles that prohibit configurations and also obstruct visibility. The only assumption made about the target is that its motions are continuous. Obviously, the motion strategy must ensure that each portion of the workspace is in view at some point in time; however, the more difficult task is to prevent the target from "sneaking" into a region that has already been explored.

Several applications can be envisioned for problems and motion strategies of this type. For example, suppose a building security system involves a few mobile robots with cameras or range sensors that can detect an intruder. Stationary or limited degree-of-freedom camera bases could also be installed. A patrolling route can be automatically computed that guarantees that any mobile intruder will eventually be found. To optimize expenses, it would also be important to know the minimum number of robots that would be needed. Applications are not necessarily limited to adversarial targets. For example, the task might be to automatically locate another mobile robot, items in a warehouse or factory that might get moved during the search process, or possibly even people in a search/rescue effort. Such strategies could be used by automated systems or by human searchers.

Since the task is to *guarantee* that the target is found for all possible target motions, worst-case analysis will naturally be considered for modeling the target. In the analysis, the target will thus be termed an *evader*, although in an application the actual target might not be adversarial. Likewise, the robots that are equipped with vision or range sensing are termed *pursuers*. The pursuers and evader are modeled as points in the plane (alternatively general configuration-space representations could be used [10], but only 2-D configuration spaces are addressed in this paper), and only continuous motions are permitted. Two interesting research issues follow from this general problem: 1) What bounds can be established on the number of pursuers needed to solve the problem, expressed in terms of the geometric and topological complexity of the free space? 2) Can a successful solution strategy be efficiently computed for a given problem? Our current progress on these two topics is discussed in this paper.

The general problem is an extension or combination of problems that have been considered in several contexts. Pursuit-evasion scenarios have arisen in a variety of applications such as air traffic control, military strategy, and trajectory tracking. This has resulted in the formal study of general decision problems in which two decision makers have diametrically opposing interests. Classical pursuit-evasion games express differential motion models for two opponents, and conditions of capture or optimal strategies are sought [8]. For example, in the classical Homicidal Chauffeur game, conditions of inevitable collision can be expressed in terms of the nonholonomic turning-radius constraints of the pursuer and evader. Although interesting decision problems arise through the differential motion models, geometric free-space constraints are usually not considered in classical pursuit-evasion games. Once these constraints are introduced, the problem inherits the additional complications that arise in geometric motion planning.

A region of capture is often associated with a pursuit-evasion problem, and the "capture" for our problem is defined as having the evader lie within a line-of-sight view from a pursuer. Several interesting results have been obtained for pursuit-evasion in a graph [12, 14]. However, the visibility polygon along with motions in a free space add geometric information that must be utilized, and also leads to connections with the static art gallery problems [13]. In the limiting case, art gallery results serve as a loose upper bound on the number of pursuers by allowing a covering of the free space by static guards, guaranteeing that any evader will be immediately visible. Far fewer guards are needed when they are allowed to move and search for an evader; however, the required motion strategies can become complicated.

2 Problem Definition

The pursuers and evader are modeled as points that translate in a 2-D bounded, Euclidean workspace that contains polygonal obstacles. See Figure 1 for illustrative examples. Let F represent the closure of the collision-free space (referred to as $\mathcal{C}_{valid}$ in [10]). All pursuer and evader positions must lie in F. Let $e(t) \in F$ denote the position of the *evader* at time $t \geq 0$. It is assumed that $e : [0, \infty) \rightarrow F$ is a continuous function, and that the evader is capable of executing arbitrarily fast motions. The initial position $e(0)$ and the path e are assumed unknown to pursuers.

Let $\gamma^i(t)$ denote the position of the i^{th} *pursuer* at time $t \geq 0$. Let γ^i represent a continuous path of the i^{th} pursuer of the form $\gamma^i : [0, \infty) \rightarrow F$. Let γ denote a *(motion) strategy*, which refers to a specification of a continuous path for every pursuer: $\gamma = \{\gamma^1, \ldots, \gamma^N\}$ for N pursuers.

For any point, $q \in F$, let $V(q)$ denote the set of all points in F that are visible from q (i.e., the linear segment joining q and any point in $V(q)$ lies in F). A strategy, γ, is a *solution strategy* if for every continuous function $e : [0, \infty) \rightarrow F$ there exists a time $t \in [0, \infty)$ and an $i \in \{1, \ldots, N\}$ such that $e(t) \in V(\gamma^i(t))$. This implies that the evader will eventually be seen by one or more pursuers, regardless of its path. Let $H(F)$ represent the minimum number of pursuers for which there exists a solution strategy for F.

Two basic problems are considered:

1. Determine $H(F)$

2. For a given F, find a solution strategy, γ, using $H(F)$ pursuers

3 How Many Pursuers are Needed?

The minimum number of pursuers, $H(F)$, required to find an evader in a given free space F generally depends on both the geometry and topology of the free space. For example, $H(F) \geq 2$ when F is multiply-connected (the evader can always hide behind a hole to avoid being seen by a single pursuer). Figure 1 shows examples that have subtle differences; however, $H(F)$ varies. In [15] a class of simple polygons called "hedgehogs" is identified for which a single pursuer suffices.

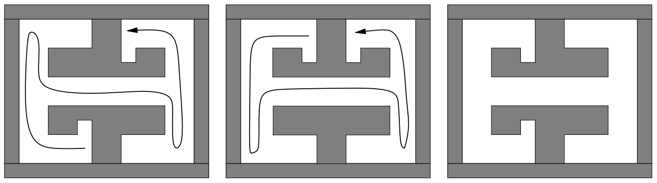
Figure 1. The rightmost example requires two pursuers, whereas the others require only one.

We have derived several bounds on $H(F)$ over certain classes of free spaces. It can be shown that for any simply-connected free space F with n edges, at worst $H(F) = O(\lg n)$, and for general free spaces with h holes, at worst $H(F) = O(h + \lg n)$ [6]. To obtain the first result, F can be recursively decomposed by placing pursuers on partitioning edges to prevent the evader from moving from one portion of F to another. A logarithmic number of pursuers can be systematically swept across partition edges to obtain the solution strategy. To obtain the second result, a linear number of line segments can be used to connect between holes and the exterior edges of F so that any continuous path that is not homotopic to a stationary path must cross one of the line segments. One pursuer can be placed on each segment to effectively reduce the problem to that of a simply-connected free space.

Lower bounds can also established which indicate problems that require at least some number of pursuers. To construct difficult problem instances, a well-studied problem from graph theory can be used. Let *Parsons' problem* refer to the graph-searching problem presented in [12, 14]. The task is to specify the number of pursuers required to find an evader that can execute continuous motions along the edges of a graph. Instead of using visibility, capture is achieved when one of the pursuers "touches" the evader. We have shown that for any instance of Parsons' problem on a planar graph, there exists an equivalent geometric instance [6]. The basic idea is to replace each edge in the graph by a thin corridor that has four bends. The key difference between the graph problem and the geometric problem is the power of visibility, which is essentially removed once four-bend corridors are used. By transforming difficult graph instances into geometric instances, it can be shown that exploring a tree of corridors of the type shown in Figure 2 requires $k+1$ pursuers in which k is the height of the tree (thus establishing $H(F) = \Omega(\lg n)$) [6]. Examples can also be constructed that establish $H(F) = \Omega(\sqrt{h} + \lg n)$.

4 Computing a Solution Strategy

4.1 General issues

In general, one would prefer a *complete* algorithm, which must compute a solution strategy for a given number of pursuers, if such a strategy exists. It is natural to compare the notion of completeness for this problem to completeness for the basic motion planning problem (i.e., the algorithm will find a collision-free path if such

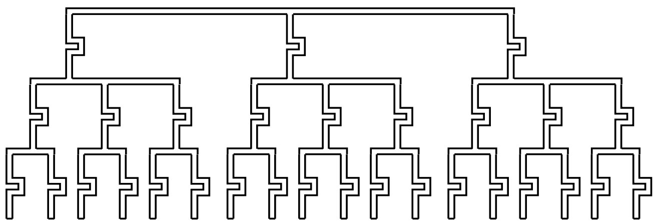

Figure 2. A ternary tree of bent corridors requires one pursuer per level (for this example $H(F) = 4$).

a path exists [3]). One important difference, however, is that the *minimum* number of pursuers is crucial, but does not have a correspondence for the basic path planning problem. A variety of simple, heuristic algorithms can be developed that require more pursuers than necessary (for example, triangulate the workspace, and place a static pursuer in each triangle). By building on some results from graph theory, it can be shown that the general problem of determining $H(F)$ for a polygonal environment is NP-hard [6]. The solutions can also be quite complicated (we have found examples that require clearing the same region $\Omega(n)$ times for n edges).

Because the position of the evader is unknown, one does not have direct access to the state at a given time. This motivates the consideration of an information space that identifies all unique situations that can occur during the execution of a motion strategy. Let a *state space*, X, be spanned by the coordinates $x = (x^1, \ldots, x^N, x^e)$, in which x^i for $1 \leq i \leq N$ represents the position of the i^{th} pursuer, and x^e represents the position of the evader. Since the positions of the pursuers are always known, let X^p denote the subspace of X that is spanned by the pursuer positions, $x^p = (x^1, \ldots, x^N)$.

It will be useful to analyze a strategy in terms of manipulating the set of possible positions of the evader. Any region in F that might contain the evader will be referred to as *contaminated*, otherwise it will be referred to as *cleared*. Let $S \subseteq F$ represent the set of all contaminated points in F. Let $\eta = (x^p, S)$ for which $x^p \in X^p$ and $S \subseteq F$ represent an *information state*. The information space is a standard representational tool for problems that have imperfect state information, and has been useful in optimal control and dynamic game theory (e.g., [1]), and in motion planning [2, 4, 11].

For a fixed strategy, γ, a path in the information space will be obtained by $\eta(t) = (\gamma^1, \ldots, \gamma^N, S(t))$ in which $S(t)$ can be determined from an initial $S(0)$ and the trajectories $\{\gamma^i(t') | t' \in [0, t]\}$ for each $i \in \{1, \ldots, N\}$.

We next describe a general mechanism for defining critical information changes. This is inspired in part by a standard approach used in motion planning, which is to preserve completeness by using a decomposition of the configuration space that is constructed by analyzing critical events. The next definition describes an information invariance property, which allows the information space to be partitioned into equivalence classes. A connected set $D \subseteq X^p$ is *conservative* if $\forall \eta$ such that $x^p \in D$, and $\forall \gamma : [t_0, t_1] \rightarrow D$ such that γ is continuous and $\gamma(t_0) = \gamma(t_1) = x^p$, then the same information state is obtained. This definition implies path invariance within a conservative cell [6]. Just as in the case of motion planning algorithms based on critical curves and noncritical regions [10], one can only consider sequences of cells in the search for a strategy while maintaining completeness. In our case, however, these cells exist in the information space.

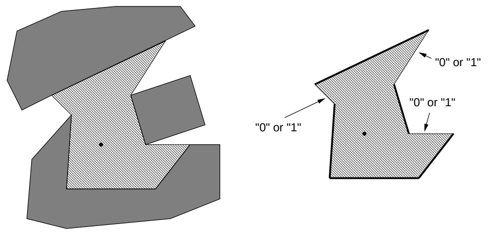

Figure 3. Edge labels can be used to encode the information state.

4.2 The complete algorithm for one pursuer

A conservative cell decomposition will be described that is based on critical changes in edge visibility. Suppose the pursuer is at a point $q \in F$. Consider the circular sequence of edges in the resulting visibility polygon. The edges generally alternate between bordering an obstacle and bordering free space. See Figure 3. Let each edge that borders free space be referred to as a *gap edge*. Consider associating a binary label with each gap edge. If the portion of the free space that borders the gap edge is contaminated, then it is assigned a "1" label; otherwise, it is assigned a "0" label indicating that it is clear. Let $B(q)$ denote a binary sequence that corresponds to labelings that were assigned from $q \in F$. Note that the set of all contaminated points is bounded by a polygon that must contain either edges of F or gap edges from the visibility polygon of the pursuer. Thus, the specification of q and $B(q)$ uniquely characterizes the information state.

Consider representing the information state using q and $B(q)$, and let pursuer move in a continuous, closed-loop path that does not cause gap edges to appear or disappear at any time. Each gap edge will continuously change during the motion of the pursuer; however, the corresponding gap edge label will not change. The information state cannot change unless gap edges appear or disappear. For example, consider the problem shown in Figure 4 which shows a single pursuer that is approaching the end of a corridor. If the closed-loop motion on the left is executed, the end of the corridor remains contaminated. This implies that although the information state changes during the motion, the original information state is obtained upon returning. During the closed-loop motion on the right, the gap edge disappears and reappears. In this case, the resulting information state is different. The gap label is changed from "1" to "0".

Hence, a cell decomposition that maintains the same corresponding gap edges will only contain conservative cells. The idea is to partition the free space into convex cells by identifying critical places at which edge visibility changes. A decomposition of this type has been used for robot localization in [7, 16], and generates $O(n^3)$ cells in

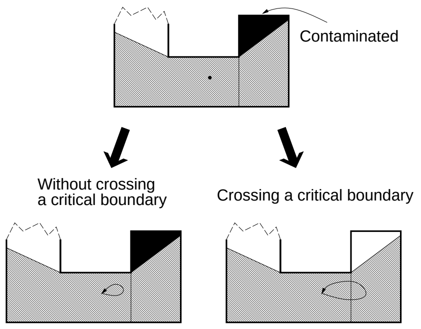
Contaminated
Without crossing a critical boundary
Crossing a critical boundary

Figure 4. A critical event in the information space can only occur when edge visibility changes.

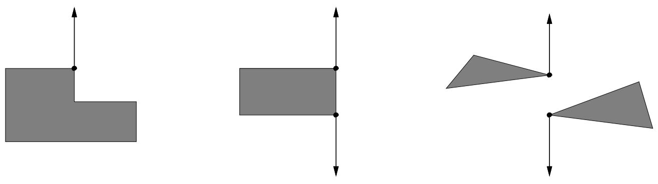

Figure 5. Ray shooting is performed for three general cases to form the edge-visibility cells.

the worst case for a simple polygon (which is always true if $H(F) = 1$). The free space can be sufficiently partitioned in our case by extending rays in the three general cases shown in Figure 5. Obstacle edges are extended in either direction, or both directions if possible. Pairs of vertices are extended outward only if both directions are free along the line drawn through the pair of points. This precludes the case in which one direction is cannot be extended; although edge visibility actually changes for this case, it does not represent a critical change in information. Our implementation uses the quad-edge structure from [5] to efficiently maintain the topological ordering of the conservative cells. Figure 7.a shows a computed example of the cell decomposition.

The next issue is searching the information space for a solution, which corresponds to specifying a sequence of adjacent cells. The solution strategy must take the form of a path that maps into F. This can be constructed by concatenating linear path segments, in which each segment connects the centroids of a consecutive pair of cells in the sequence.

The cells and their natural adjacency relationships define a finite, planar graph, G_c, referred to as the *cell graph*. Vertices in G_c are generally visited multiple times in a solution sequence because of the changing information states. For each vertex in G_c, a point, $q \in F$, in the corresponding cell can be identified, and the labels $B(q)$ can be distinct at each visit. Initially, the pursuer will be in some position at which all gap labels are "1". The goal is to find any sequence of cells in G_c that leave the pursuer at some position at which all gap labels are "0".

A directed *information state graph*, G_I, can be derived from G_c, for which each vertex is visited at most once during the execution of a solution strategy. For each vertex in G_c, a set of vertices are included in G_I for each possible $B(q)$. For example, suppose a vertex in G_c represents some cell D, and there are 2 gap edges for $B(q)$ and any $q \in D$. Four vertices will be included in G_I that all correspond to the pursuer at cell D; however, each vertex represents a unique possibility for $B(q)$: "00", "01", "10", or "11". Let a vertex in G_I be identified by specifying the pair $(q, B(q))$.

To complete the construction of G_I, the set of edges must be defined. This requires determining the appropriate gap labels as the pursuer changes cells. Suppose the pursuer moves from $q_i \in D_i$ to $q_j \in D_j$. For the simple case shown in the lower right of Figure 4, assume that the gap edge on the left initially has a label of "0" and the gap edge on the right has a label of "1". Let the first bit denote the leftmost gap edge label. The first transition is from "01" to "0", and the second transition is from "0" to "00". The directed edges in G_I are $(q_i,\text{"01"})$ leads to $(q_j,\text{"0"})$, $(q_j,\text{"0"})$ leads to $(q_i,\text{"00"})$.

In the case of multiple gap edges, correspondences must be determined to correctly compute the gap labels. Consider the example shown in Figure 6 which illustrates the general cases that can occur. A gap edge from $V(q_i)$ corresponds to a gap edge from $V(q_j)$ if they share a vertex, and neither touch the extension of their common cell boundary. This case is shown in the upper left of Figure 6. In this case the binary label with be preserved when traveling directly from q_1 to q_2. The case is more interesting when gap edges touch the extension of the cell boundary, as in the lower portion of Figure 6. In general, all edges that touch the extension below the cell correspond to each other, and all edges that touch the extension above the cell separately correspond to each other. Transitions of this type essentially cause gap edges to be split or merged. There are two gap edges in the lower portion of Figure 6 while the pursuer is at q_1; however, there is only one gap edge when the pursuer is at q_2. In the transition from q_1 to q_2, if the gap edges at q_1 are labeled "0" and "0", then the corresponding gap from q_2 will be labeled "0". If either gap edge at q_1 is labeled "1", then the gap edge label from q_2 will be "1" (contamination spreads easily). In general, if any n gap edges are merged, the corresponding gap edges will receive a "1" label if any of the original gap edges contain a "1" label.

Once the gap edge correspondences have been determined, the information state graph can be searched using Dijkstra's algorithm with an edge cost that corresponds to the distance traveled in the free space by the pursuer. Unfortunately, the precise complexity of the complete algorithm cannot be determined because it is still open whether the problem even lies in NP. In the worst-case, examples can be constructed that yield an exponential number of information states, but it is not clear whether these information states necessarily have to be represented and searched to determine a solution (or to verify a solution).

4.3 Coordinating multiple pursuers

In general, the conservative cell concept can be applied to yield a cell decomposition of X^p, which is the $2N$-dimensional space that encodes the positions of the pursuers; however, some of the cell boundaries are alge-

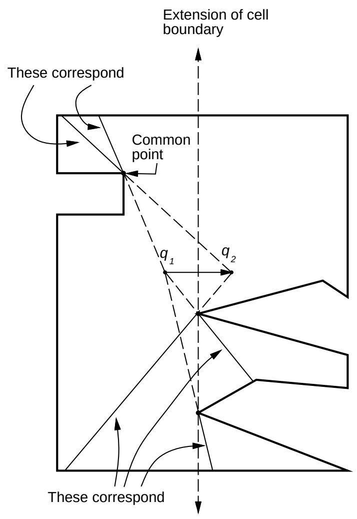

Figure 6. The correspondences between gap edges from different neighboring cells can be directly determined. The information states are updated when moving between cells by using this correspondence.

braic manifolds. The algebraic constraints significantly increase the implementation difficulty and add numerous cells which decreases practical efficiency.

In Section 5 we show some examples that were computed by coordinating multiple pursuers. We have developed a decoupled planning approach (losing completeness), which is inspired by typical approaches to multiple robot planning problems [10]. Suppose a problem cannot be solved by a single pursuer. The first step is to have the pursuer clear as much area as possible and stop. The fixed pursuer's visibility polygon partitions the free space into components that can each by explored as a separate subproblem using the complete algorithm for a single pursuer. If each component can be solved by a single pursuer, then only two pursuers are needed in total (the same pursuer can be used for each component). In general, this type of search can be repeated recursively to coordinate more pursuers, until the problem is solved. In many cases, pursuers that are fixed during the clearing of one portion of the free space can eventually be moved to assist in another portion, further reducing the number of pursuers.

5 Computed examples

The algorithm is implemented in C++ and was executed on an SGI Indigo2 workstation with a 200 Mhz MIPS R4400 processor. Four computed examples are shown in Figures 7-10. The total computation times for these examples were 1.02, 0.21, 1.54, and 4.12 seconds, respectively. The first example could be solved with a single pursuer, while the other examples required coordinating multiple pursuers. Figure 7 shows both the cell decomposition and the snapshots along the computed solution trajectory. Figure 8 corresponds to one of the corridor trees described in Section 3. The first pursuer waits at the junction while the second pursuer clears the remaining two components. Figure 9 requires two pur-

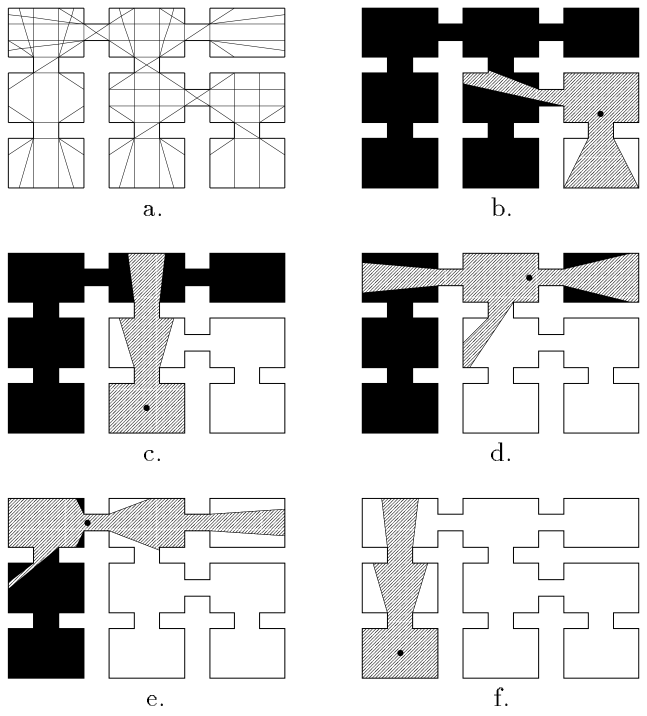

Figure 7. a) The conservative cell decomposition; b-f) Snapshots along the computed solution trajectory. In the final snapshot, there is no place remaining where the evader could be hiding. Black represents the contaminated region, white represents the cleared region, and gray represents the current visibility region.

suers, and was solved by halting the first pursuer near the center hole (as shown in Figure 9.b) while the other pursuer cleared the remaining components. Figure 10 represents a very challenging example, which was solved using only two pursuers.

6 Discussion

A visibility-based motion planning problem has been identified in this paper that involves searching for an unpredictable target in a workspace that contains obstacles. Several bounds on the minimum number of needed pursuers were discussed. A general decomposition concept based on information *conservative* cells was introduced, which led to a efficient, complete algorithm for $H(F) = 1$ that has been implemented and tested. The algorithm was then augmented to coordinate multiple pursuers; this extension solves many problems, but does not generally yield the optimal number of pursuers.

Several variations and extensions of the problem are worth exploring. In addition to a visibility region, each pursuer can have a region of capture, and the task can be to capture the evader using one or more pursuers. Using the current evader model, only connectivity issues become critical for determining a solution strategy; however, the problem can be made more challenging by strengthening the model to include a bounded velocity, or possibly stochastic prediction. The topological issues

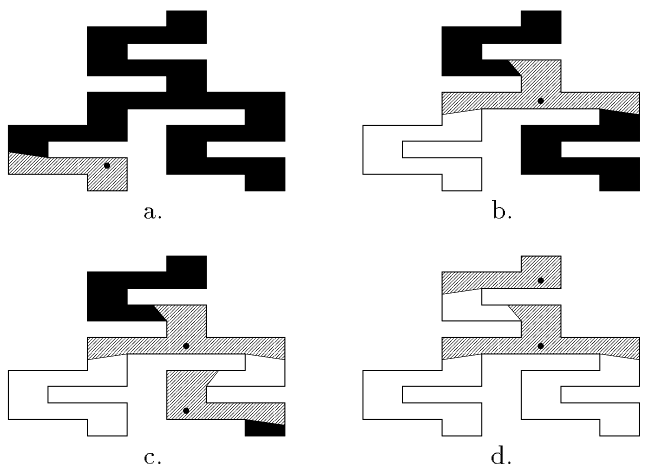

Figure 8. This simply-connected free space requires two pursuers. The first pursuer stops at the junction.

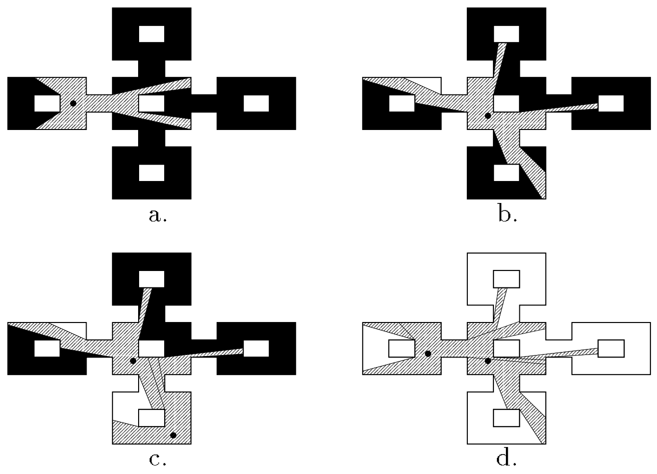

Figure 9. This example requires two pursuers.

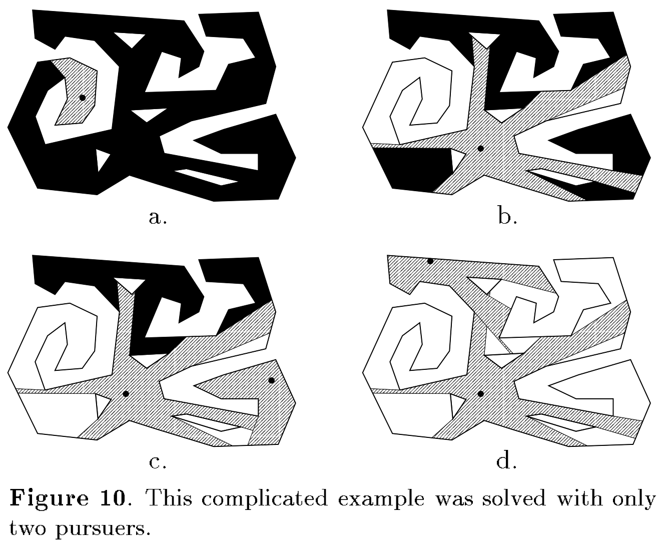

Figure 10. This complicated example was solved with only two pursuers.

could become significantly more complex for 3-D free spaces. Many vision systems have a limited field of view, and our problem can be adapted to planning strategies that sweep viewing angles in addition to moving the pursuers. Finally, a cost functional could be additionally defined, leading to problems such as finding the evader in minimum time.

Acknowledgments

This work is funded by ARO MURI grant DAAH04-96-1-007 and ONR grant N00014-94-1-0721. We thank Julien Basch, Frédéric Cazals, Bruce Donald, Héctor González-Baños, and Li Zhang, for their helpful suggestions.

References

[1] T. Başar and G. J. Olsder. *Dynamic Noncooperative Game Theory.* Academic Press, London, 1982.

[2] J. Barraquand and P. Ferbach. Motion planning with uncertainty: The information space approach. In *IEEE Int. Conf. Robot. & Autom.*, pages 1341–1348, 1995.

[3] J. F. Canny. *The Complexity of Robot Motion Planning.* MIT Press, Cambridge, MA, 1988.

[4] M. Erdmann. Randomization for robot tasks: Using dynamic programming in the space of knowledge states. *Algorithmica*, 10:248–291, 1993.

[5] L. Guibas and J. Stolfe. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *AMC Trans. Graphics*, 4(2):74–123, 1985.

[6] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani. A visibility-based pursuit-evasion problem. 1996. Submitted to *International Journal of Computational Geometry and Applications.*

[7] L. J. Guibas, R. Motwani, and P. Raghavan. The robot localization problem. In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *Proc. 1st Workshop on Algorithmic Foundations of Robotics*, pages 269–282. A.K. Peters, Wellesley, MA, 1995.

[8] R. Isaacs. *Differential Games.* Wiley, New York, NY, 1965.

[9] A. S. Lapaugh. Recontamination does not help to search a graph. *J. ACM*, 40(2):224–245, April 1993.

[10] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, MA, 1991.

[11] S. M. LaValle. *A Game-Theoretic Framework for Robot Motion Planning.* PhD thesis, University of Illinois, Urbana, IL, July 1995.

[12] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou. The complexity of searching a graph. *J. ACM*, 35(1):18–44, January 1988.

[13] J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, New York, NY, 1987.

[14] T. D. Parsons. Pursuit-evasion in a graph. In Y. Alani and D. R. Lick, editors, *Theory and Applcation of Graphs*, pages 426–441. Springer-Verlag, Berlin, 1976.

[15] I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM J. Comput.*, 21(5):863–888, October 1992.

[16] R. Talluri and J. K. Aggarwal. Mobile robot self-location using model-image feature correspondence. *IEEE Trans. Robot. & Autom.*, 12(1):63–77, February 1996.