Visibility-Based Pursuit-Evasion in a Polygonal Environment

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

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

Abstract. *This paper addresses the problem of planning the motion of one or more pursuers in a polygonal environment to eventually "see" an evader that is unpredictable, has unknown initial position, and is capable of moving arbitrarily fast. This problem was first introduced by Suzuki and Yamashita. Our study of this problem is motivated in part by robotics applications, such as surveillance with a mobile robot equipped with a camera that must find a moving target in a cluttered workspace.*
A few bounds are introduced, and a complete algorithm is presented for computing a successful motion strategy for a single pursuer. For simply-connected free spaces, it is shown that the minimum number of pursuers required is $\Theta(\lg n)$. *For multiply-connected free spaces, the bound is* $\Theta(\sqrt{h} + \lg n)$ *pursuers for a polygon that has* n *edges and* h *holes. A set of problems that are solvable by a single pursuer and require a linear number of recontaminations is shown. The complete algorithm searches a finite cell complex that is constructed on the basis of critical information changes. It has been implemented and computed examples are shown.*

1 Introduction

The general problem addressed in this paper is an extension or combination of problems that have been considered in several contexts. Interesting results have been obtained for pursuit-evasion in a graph, in which the pursuers and evader can move from vertex to vertex until eventually a pursuer and evader lie in the same vertex [16, 19]. The *search number* of a graph refers to the minimum number of pursers needed to solve a pursuit-evasion problem, and has been closely related to other graph properties such as cutwidth [15, 17]. Pursuit-evasion scenarios in continuous spaces have arisen in a variety of applications such as air traffic control [1], military strategy [10], and trajectory tracking [9]. 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. A moving visibility polygon in a polygonal environment adds geometric information that must be utilized, and also leads to connections with the static art gallery problems [18, 21]. 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 quite complex. A closely related art gallery variant is the watchman tour problem [4]. In this case a minimum-length closed path is computed such that any point in the polygon is visible from some point along the path. In our case, however, the pursuers have the additional burden of ensuring that an evader cannot "sneak" to a portion of the environment that has already been explored. The problem that we consider and other variations have been considered previously in [5, 22]. It was stated in [21] that it remained an interesting challenge to determine if a polygon is searchable by a single pursuer.

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. 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.

2 Problem Definition

The pursuers and evader are modeled as points that translate in a polygonal free space, 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 the evader is capable of moving arbitrarily fast. The initial position $e(0)$ and the path e are assumed unknown to pursuers. Any region in F that might contain the evader will be referred to as *contaminated*, otherwise it will be referred to as *cleared*. If a region is contaminated, becomes cleared, and then becomes contaminated again, it will be referred to as *recontaminated*.

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 the specification of a continuous path for every pursuer: $\gamma = \{\gamma^1, \ldots, \gamma^N\}$.

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.

Section 3 presents some bounds on $H(F)$ for classes of free spaces, and also shows that some polygons for which $H(F) = 1$ only admit solutions that require a linear number of recontaminations. Section 4 addresses the problem of computing a solution strategy, γ, for a given F.

3 Worst-Case Bounds

Several new bounds are presented in this section. For a simply-connected free space, F, with n edges, it is shown that $H(F) = \Theta(\lg n)$. For a free space, F, with h holes, it is shown that $H(F) = \Omega(\sqrt{h} + \lg n)$ and $H(F) = O(h + \lg n)$. For the class of problems in which $H(F) = 1$, it is shown that the same region can require recontamination as many as $\Omega(n)$ times. This result is surprising because pursuit-evasion in a graph is known not to require any recontaminations [11].

Consider the problem of determining the minimum number of pursuers, $H(F)$, required to find an evader in a given free space F. This number will generally depend on both the topological and geometric complexity of F. In [22] a class of simple polygons is identified for which a single pursuer suffices (referred to as "hedgehogs"). For any F that has at least one hole, it is clear that at least two pursuers will be necessary; if a single pursuer is used, the evader could always move so that the hole is between the evader and pursuer. In some cases subtle changes in the geometry significantly affect $H(F)$.

Consider $H(F)$ for the class of simply-connected free spaces. Let n represent the number of edges in the free space, which is represented by a simple polygon in this case. A logarithmic worst-case bound can be established:

Theorem 1. *For any simply-connected free space F at worst $H(F) = O(\lg n)$.*

Proof: The proof is built on the following observation. Suppose that two vertices of F are connected by a linear segment, thus partitioning F into two simply-connected, polygonal components, F_1 and F_2. If $H(F_1) \leq k$ and $H(F_2) \leq k$ for some k, then $H(F) \leq k + 1$ because the same k pursuers can be used to clear both F_1 and F_2. This requires placing a static $(k + 1)^{th}$ pursuer at the edge common to F_1 and F_2 to keep F_1 cleared after the k pursuers move to F_2 (assuming arbitrarily that F_1 is cleared first).

In general, if two simply-connected polygonal regions share a common edge and can each be cleared by at most k pursuers, then the combined region can be cleared at most $k + 1$ pursuers. Recall that for any simple polygon, a pair of vertices can always be connected so that polygon is partitioned into two regions, each with at least one third of the edges of the original polygon [3]. This implies that F can be recursively partitioned until a triangulation is constructed, and each triangular region only requires $O(\lg n)$ recombinations before F is obtained (i.e., the recursion depth is logarithmic in n). Based on the previous observation and the fact that each triangular region can be trivially searched by a single pursuer, $H(F) = O(\lg n)$. □

The remaining question for simply-connected free spaces is whether there actually exist problems that require a logarithmic number of pursuers. Some results from graph searching will first be described and utilized to construct difficult worst-case problem instances. Let *Parsons' problem* refer to the graph-searching problem presented in [16, 19]. 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. Let G represent a graph, and $S(G)$ represent the number of needed pursuers, referred to as the search number of G.

The following lemma implies that a geometric realization of any planar graph instance can be constructed:

Lemma 2. *For every planar graph, G, there exists a polygonal free space F such that Parsons' problem on G is equivalent to the visibility-based pursuit evasion problem on F.*

Proof: Consider a planar representation of G in which linear segments correspond to the edges. Each linear segment can be replaced by a corridor of sufficient length as shown in Figure 1, with $\epsilon > 0$ chosen such that no pair or corridors intersect. Portions of the corridor edges can be removed at corridor junctions to prevent overlap.

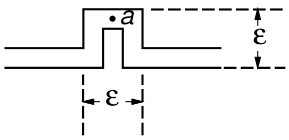

Fig. 1. Second-order visibility between the two entrances is not maintained.

It remains to establish that searching the environment constructed by the network of bent corridors is equivalent to searching a planar graph. If a solution strategy for Parsons' problem is specified as an ordered set of traversed edges for each pursuer, then traversing the corresponding corridors will clearly solve the geometric problem. Consider a given solution for the geometric problem. If the pursuers move from junction to junction without reversing within a corridor, then traversing the equivalent edges also solves the planar graph problem. For the geometric problem, a static pursuer can be placed in a corridor (as in position a in Figure 1), allowing two other pursuers to clear the corridor from each end without leaving the junctions unguarded. In this case and in similar cases, the pursuer that is fixed in the geometric problem can clear the corresponding edge by traveling between its endpoints in G; thus, the graph problem can still be solved using the same number of pursuers. □

This theorem, from [19], is useful for proving Theorem 4:

Lemma 3. (Parsons) *Let G be a tree. Then $S(G) = N + 1$ if and only if there exists a vertex in G whose removal separates G into three components, G_1, G_2, and G_3, such that $S(G_i) \geq N$ for $i \in \{1, 2, 3\}$.*

Theorem 4. *There exist simply-connected free spaces F with n edges such that $H(F) = \Omega(\lg n)$.*

Proof: Using Lemma 3, a tree, G, can be constructed recursively that has a constant branching factor of three, height $N - 1$, and requires N pursuers (an example is given in [19]). By Lemma 2, an equivalent geometric instance can be constructed for any N. □

Figure 2.a shows an example, which relies on Lemma 3. Theorem 1 and Theorem 4 together imply a tight logarithmic bound, $H(F) = \Theta(\lg n)$.

Next consider the class of problems for which F has h holes.

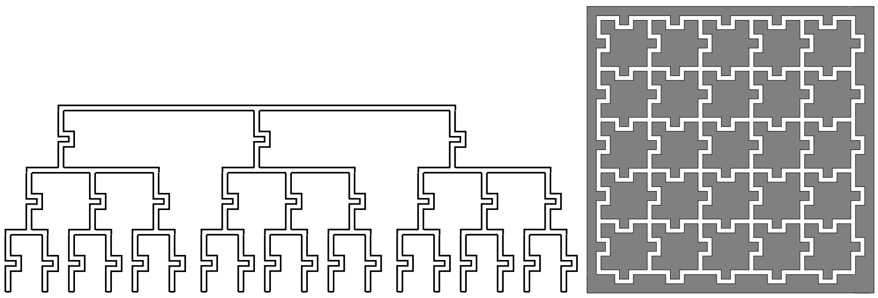

Fig. 2. a) An instance from a sequence of simply-connected free spaces that require $\Omega(\lg n)$ pursuers. b) An instance from a sequence of problems that requires a number of pursuers that is at least proportional to the square root of the number of holes.

Theorem 5. *For any free space F with h holes at worst $H(F) = O(\sqrt{h} + \lg n)$.*

Proof: Divide the pursuers into two groups: $O(\sqrt{h})$ pursuers will be used to reduce the polygon to simply-connected components, and $O(\lg n)$ pursuers will be used to clear each component. Construct an arbitrary triangulation of F. Let a *trichromatic triangle* be defined as a triangle that touches three distinct connected components of the boundary of F. Using at most $O(h)$ trichromatic triangles, F can be partitioned into simply-connected components whose boundaries are comprised of the boundary of F and edges of trichromatic triangles. To establish this, form a planar graph by placing a vertex in each hole of F and one vertex outside of F. For every trichromatic triangle edge joining two boundaries, form an edge for this graph by joining the vertices corresponding to these boundaries by a path along the trichromatic edge, in the obvious way. From the planarity of this graph we can easily argue that the overall number of trichromatic edges, and therefore of trichromatic triangles, is $O(h)$.

Consider the dual graph of the triangulation, which has $O(n)$ edges and vertices. Take the subgraph induced by taking only the vertices that correspond to trichromatic triangles in the original triangulation. The planar graph separator theorem [14] implies that at most $O(\sqrt{h})$ edges can be chosen to partition the graph into two portions with at least one third of the edges on each side of the partition. Each edge in the induced subgraph corresponds to a simply-connected region of F that can be cleared with $O(\lg n)$ pursuers by Theorem 1. In fact, the same set of pursuers can be used for each simply connected component. The $O(\sqrt{h})$ pursuers then form a barrier that maintains the cleared areas obtained by other pursuers on either side, much in the same way as the partitioning edges in the proof of Theorem 1. The planar graph separator theorem can be applied recursively to the remaining portions of F on either side of a barrier, and the free space can be cleared using the same progression as for Theorem 1. At the i^{th} level of recursion, at most $\frac{2}{3}$ as many pursuers will be needed to form a barrier in comparison to the $(i-1)^{th}$ level of recursion. The free space is reduced to simply-connected components that can be cleared using $O(\lg n)$ pursuers. The total number of pursuers needed to form barriers is $O(\sqrt{h} + \sqrt{\frac{2}{3}h} + \sqrt{\frac{4}{9}h} + \cdots) = O(\sqrt{h})$. Thus, F can be cleared using at most $O(\sqrt{h} + \lg n)$ pursuers. □

Figure 2.b shows an illustrative example of the following theorem:

Theorem 6. *There exist free spaces F with h holes such that $H(F) = \Omega(\sqrt{h} + \lg n)$.*

Proof: For any positive integer k, a planar graph of cutwidth k can be constructed using $O(k^2)$ vertices and edges. Recall that the cutwidth, $CW(G)$, is the minimum cutwidth taken over all possible linear layouts of G. A linear layout of G is a one-to-one function mapping the vertices of G to integers, and the cutwidth for a particular layout is the maximum over all i of the number of edges connecting vertices assigned to integers less than i to vertices assigned to integers as large as i. Define a sequence of planar graphs, $G_1, G_2, \ldots$. Let the vertices of G_k correspond to the set of all points with integer coordinates, (i,j), such that $0 \le i,j \le k$. Let the edges of G_k connect any two vertices for which one coordinate differs by one unit (i.e., a standard four-neighborhood). The cutwidth of G_k is k.

It is established in [15] that for all graphs G, the search number $S(G)$ is related to the cutwidth as $S(G) \le CW(G) \le \lfloor deg(G)/2 \rfloor \cdot S(G)$, in which $deg(G)$ is the maximum vertex degree of G. Since $deg(G_k) = 4$, $S(G_k) \le k \le 2S(G_k)$. Using Lemma 2, geometric instances of G_k can be constructed. Both G_k and each geometric instance require $\Omega(k)$ pursuers. There is a quadratic number of holes in each geometric instance; hence, $H(F) = \Omega(\sqrt{h})$. This corridor structure can be combined with the structure from Theorem 4 to yield an example that requires $\Omega(\sqrt{h} + \lg n)$ pursuers. □

Theorem 6 and Theorem 5 together imply a tight bound, $H(F) = \Theta(\sqrt{h} + \lg n)$.

The final theorem of this section pertains to the class of free spaces that can be searched by a single pursuer. A similar result is also obtained in [5]. It states that there exist examples that require recontaminating some portion of the free space a linear number of times. This result is surprising because for Parsons' problem it was shown in [11] that no recontamination is necessary (a shorter proof of this appears in [2]). In [22] a free space was given that requires two recontaminations, which at least established that recontamination is generally necessary for visibility-based pursuit evasion. Theorem 7 establishes that a linear number of recontaminations can be needed, and it still remains open to determine whether the number of recontaminations can be bounded from above by a polynomial, which would imply that the problem of deciding whether $H(F) = 1$ lies in NP.

Theorem 7. *There exists a sequence of simply-connected free spaces with $H(F) = 1$ such that $\Omega(n)$ recontaminations are required for n edges.*

Proof: It will be shown that the example in Figure 3 requires $k-2$ recontaminations by visiting the point $a \in F$ a total of $k-1$ times to repeatedly clear the "peak." Without loss of generality, consider the set of strategies that can be specified by identifying the sequence of points, $a, b_1, \ldots, b_k, c_1, \ldots, c_k$, that are visited. Assume that the shortest-distance path is taken between any pair of points. Consider visiting b_i for some $1 < i < k$, followed by a visit to another "leg", say b_j (or c_j). If any legs between b_i and b_j are contaminated, then b_i will get contaminated, which undoes previous work. If all legs are initially contaminated, then they must be visited in one of two orders: $(b_1, c_1, b_2, c_2, \ldots, b_k, c_k)$ or $(b_k, c_k, b_{k-1}, c_{k-1}, \ldots, b_1, c_1)$. Because of symmetry, consider visiting the legs from left to right without loss of generality. Assume that the peak is initially contaminated. The points b_1 and c_1 can be visited to clear the leftmost set of legs; however, these will get contaminated when b_2 is visited. By traveling from c_1 to a to b_2, the leftmost set of legs remain cleared because the peak is cleared. When c_2 is visited, the leftmost three legs remain cleared; however, the peak becomes recontaminated. Thus, a will have to be visited again before clearing b_3. By induction on i for $1 < i \leq k$, the peak will have to be cleared by visiting a each time between visits to c_i and b_{i+1}. This implies that a will be visited $k-1$ times, resulting in $k-2$ recontaminations. □

4 Computing a Solution Strategy

Section 4.1 defines a general information space (or space of knowledge states) for this problem, and provides a general method for partitioning the information space into equivalence classes, which can reduce the general problem to finite cell searching. NP-hardness is also established in Section 4.1. Section 4.2 presents a

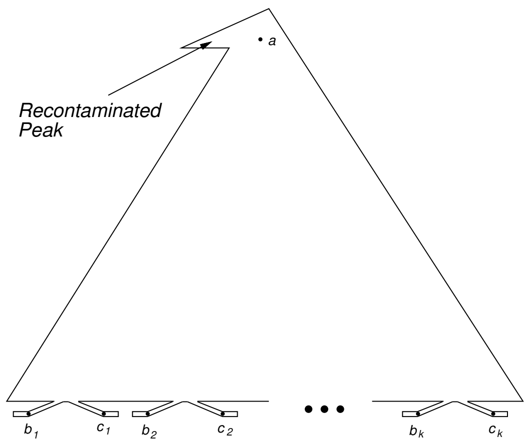

Fig. 3. A linear number of recontaminations is required. Although this polygon can be searched by a single pursuer, the peak must be visited $k-1$ times.

complete algorithm for the case in which $H(F) = 1$ that computes a solution strategy by decomposing F into convex cells based on edge visibility. This algorithm is quite efficient in practice, and was used to compute the examples shown in Section 5.

4.1 General Concepts

A *complete* algorithm 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 a path exists). 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 might use more pursuers than necessary.

The general problem is intractable if $P \neq NP$:

Theorem 8. *Computing $H(F)$ is NP-hard.*

Proof: It is shown in [17] that Parsons' problem for a planar graph with maximum vertex degree 3 is NP-complete (i.e., computing the search number, $S(G)$). By Lemma 2, equivalent geometric instances can be constructed, which implies that computing $H(F)$ is NP-hard. □

This significantly reduces hopes that an efficient algorithm can be determined for the general problem. A complete algorithm for $H(F) = 1$ is detailed in this paper, and the general techniques apply to the case in which $H(F) > 1$. In related work [13] we have developed a greedy algorithm that efficiently solves many multiple-pursuer problems.

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. 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*. Let the *information space*, $\mathcal{I}$, represent the set of possible information states. The information space is a standard representational tool for problems that have imperfect state information, and has been useful for other motion planning problems [6, 12].

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 $i \in \{1, \ldots, N\}$. Let $\Psi(\eta, \gamma, t_0, t_1)$ represent the information state that will be obtained by starting from information state η, and applying the strategy γ from t_0 to t_1. The function Ψ can be thought of as a "black box" that produces the resulting information state when a portion of a given strategy is executed.

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. For example, in [20] a cell decomposition is determined by analyzing the contact manifolds in a composite configuration space that is generated by the positions of several disks in the plane.

The next definition describes an information invariance property, which allows the information space, $\mathcal{I}$, to be partitioned into equivalence classes. A connected set $D \subseteq X^p$ is *conservative* if $\forall \eta \in \mathcal{I}$ 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, $\eta = \Psi(\eta, \gamma, t_0, t_1)$, is obtained. This implies that the information state cannot be altered by moving along closed paths in D. Just as in the case of motions in a conservative field, the following holds:

Theorem 9. (Path invariance) *If D is conservative then for any two continuous trajectories, γ_1, γ_2, mapping into D such that $\gamma_1(t_0) = \gamma_2(t_0)$ and $\gamma_1(t_1) = \gamma_2(t_1)$ then $\Psi(\eta, \gamma_1, t_0, t_1) = \Psi(\eta, \gamma_2, t_0, t_1)$, for any η.*

Proof: Select any third continuous trajectory, $\gamma_3 : [t_0, t_1] \rightarrow D$, such that $\gamma_3(t_0) = \gamma_1(t_1)$ and $\gamma_3(t_1) = \gamma_1(t_0)$ (i.e., heading in the opposite direction). Form a new trajectory, γ_{132}, by concatenating the trajectories γ_1, γ_3, and γ_2. The resulting information state will be $\Psi(\eta, \gamma_1, t_0, t_1)$ because γ_3 followed by γ_2 forms a closed-loop path, and thus yields the same information state by conservativity of D. Note that γ_1 followed by γ_3 is also a closed-loop path, which implies that γ_2 must bring the information state from η to $\Psi(\eta, \gamma_1, t_0, t_1)$. Hence, $\Psi(\eta, \gamma_1, t_0, t_1) = \Psi(\eta, \gamma_2, t_0, t_1)$. □

Thus, the information state from moving between $x_1^p \in D$ and $x_2^p \in D$ is invariant with respect to the chosen path. This partitions the information space into equivalence classes. Within is each class the particular chosen path is insignificant, which leads to a finite graph search problem.

4.2 A Complete Algorithm for a Single Pursuer

Since the general problem is NP-hard, it is worth focusing on the complete algorithm for the case of a single pursuer. The basic idea is to partition the free space into convex cells that maintain completeness, and perform a search on the resulting quotient information space. This algorithm has been implemented and tested on a variety of examples, two of which are shown in Section 5.

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. 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 a 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 [8, 23], and generates $O(n^3)$ cells in 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. Obstacle edges are extended in either direction, or both directions if possible. Pairs of vertices are extended outward

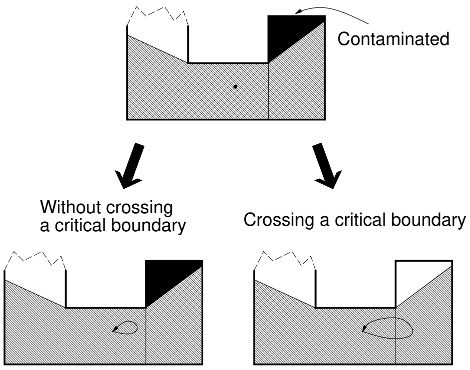

Fig. 4. A critical event in the information space can only occur when edge visibility changes.

only if both directions are free along the line drawn through the pair of points. This precludes the case in which one direction cannot be extended; although edge visibility actually changes for this case, it does not represent a critical change in information.

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 labeling of the gap edges. 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. 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. 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 (it is not even known if optimal-length solutions to the single-pursuer problem can be verified in polynomial time).

5 Computed Examples

The complete algorithm is implemented in C++ and executed on an SGI Indigo2 workstation with a 200 Mhz MIPS R4400 processor. Most problems we encountered were solved in a few seconds or less. The implementation uses the quad-edge structure from [7] to maintain the topological ordering of the conservative cells. The search strategy is Dijkstra's shortest path algorithm, in which the distance is measured from the adjacent cell centroids. Figure 5 shows two computed examples. We have also computed solutions for the example shown in Figure 3, the hookpin example described in [22] that requires two recontaminations, and several other problems.

6 Conclusions

We proved some new bounds and introduced a complete algorithm for the polygon searching problem. A logarithmic bound on the number of needed pursuers was shown for the case of simply-connected free spaces, and a square-root bound was expressed in terms of the number of holes for multiply-connected free spaces. It was also shown that there exist problems requiring a linear number of recontaminations. A few open problems remain, such as determining tight bounds on the number of pursuers for general polygons, and determining whether a polynomial-time algorithm exists to decide whether $H(F) = 1$. The complexity of our complete algorithm also remains open. It also remains an interesting pursuit to attempt to characterize the set of simple polygons such that $H(F) = 1$; interesting subsets have been characterized in [22], and our information space concepts might be useful in this endeavor.

Information space concepts were used to provide a natural characterization of the unique problem states. The visibility-based pursuit-evasion problem was

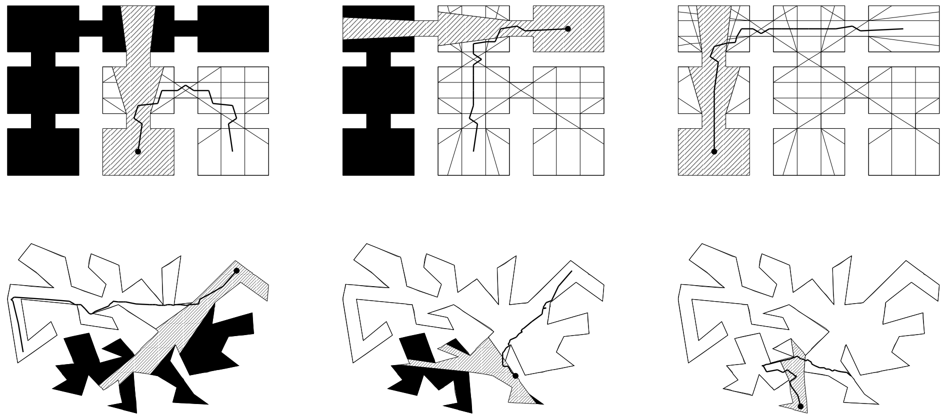

Fig. 5. Two computed examples are shown, each with three snapshots of the solution. The black area represents the contaminated region, and the white area represents the cleared region. The thick curve shows a portion of computed trajectory, which is continued in each frame. The shaded region indicates the visibility region at the final time step of the indicated portion of the trajectory. The thin lines in the cleared region indicate the cell boundaries.

established as NP-hard. The general concept of partitioning the information space on the basis of critical information changes was introduced to develop a complete algorithm. For the case in which $H(F) = 1$, the complete algorithm was implemented and tested on several examples.

References

1. T. Başar and G. J. Olsder. *Dynamic Noncooperative Game Theory.* Academic Press, London, 1982.
2. D. Bienstock and P. Seymour. Monotonicity in graph searching. *J. Algorithms*, 12:239–245, 1991.
3. B. Chazelle. A theorem on polygon cutting with applications. In *Proc. 23rd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 339–349, 1982.
4. W.-P. Chin and S. Ntafos. Optimum watchman routes. *Information Processing Letters*, 28:39–44, 1988.
5. D. Crass, I. Suzuki, and M. Yamashita. Searching for a mobile intruder in a corridor – the open edge variant of the polygon search problem. *Int. J. Comput. Geom. & Appl.*, 5(4):397–412, 1995.
6. M. Erdmann. Randomization for robot tasks: Using dynamic programming in the space of knowledge states. *Algorithmica*, 10:248–291, 1993.
7. 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.

8. 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.
9. O. Hájek. *Pursuit Games.* Academic Press, New York, 1975.
10. R. Isaacs. *Differential Games.* Wiley, New York, NY, 1965.
11. A. S. Lapaugh. Recontamination does not help to search a graph. *J. ACM*, 40(2):224–245, April 1993.
12. S. M. LaValle. *A Game-Theoretic Framework for Robot Motion Planning.* PhD thesis, University of Illinois, Urbana, IL, July 1995.
13. S. M. LaValle, D. Lin, L. J. Guibas, J.-C. Latombe, and R. Motwani. Finding an unpredictable target in a workspace with obstacles. In *Prof. IEEE Int'l Conf. on Robotics and Automation*, 1997.
14. R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM Journal of Applied Mathematics*, 36:177–189, 1979.
15. F. Makedon and I. H. Sudborough. Minimizing width in linear layouts. In *Proc. 10th ICALP, Lecture Notes in Computer Science 154*, pages 478–490. Springer-Verlag, 1983.
16. 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.
17. B. Monien and I. H. Sudborough. Min cut is NP-complete for edge weighted graphs. *Theoretical Computer Science*, 58:209–229, 1988.
18. J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, New York, NY, 1987.
19. 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.
20. J. T. Schwartz and M. Sharir. On the piano movers' problem: III. Coordinating the motion of several independent bodies. *Int. J. Robot. Res.*, 2(3):97–140, 1983.
21. T. Shermer. Recent results in art galleries. *Proc. IEEE*, 80(9):1384–1399, September 1992.
22. I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM J. Comput.*, 21(5):863–888, October 1992.
23. 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.

Acknowledgments

The research of Leonidas J. Guibas is supported by NSF grant CCR-9623851 and US Army MURI grant 5-23542-A. Rajeev Motwani's research is supported by an Alfred P. Sloan Research Fellowship, an IBM Faculty Partnership Award, an ARO MURI Grant DAAH04-96-1-007, and NSF Young Investigator Award CCR-9357849, with matching funds from IBM, Mitsubishi, Schlumberger Foundation, Shell Foundation, and Xerox Corporation. The remaining researchers are supported by ARO MURI grant DAAH04-96-1-007 and ONR grant N00014-94-1-0721. The authors thank Jian Bao, Julien Basch, Frédéric Cazals, Bruce Donald, Héctor González-Baños, Gary Kalmanovich, Jon Kleinberg, Suresh Venkatasubramanian, Li Zhang, and the anonymous reviewers, for their helpful suggestions.

This article was processed using the LaTeX macro package with LLNCS style