To appear in *Proc. of the National Conference on Artificial Intelligence (AAAI 2004)*
San Jose, California, July 2004

Visibility-based pursuit-evasion with limited field of view

Brian P. Gerkey gerkey@stanford.edu
Sebastian Thrun thrun@stanford.edu
Geoff Gordon ggordon+@cs.cmu.edu
Artificial Intelligence Lab
Stanford University
Stanford, CA 94305, USA

Abstract

We study a form of the *pursuit-evasion* problem, in which one or more searchers must move through a given environment so as to guarantee detection of any and all evaders, which can move arbitrarily fast. Our goal is to develop techniques for coordinating teams of robots to execute this task in application domains such as clearing a building, for reasons of security or safety. To this end, we introduce a new class of searcher, the ϕ-searcher, which can be readily instantiated as a physical mobile robot. We present a detailed analysis of the pursuit-evasion problem using ϕ-searchers. We show that computing the minimum number of ϕ-searchers required to search a given environment is NP-hard, and present the first complete search algorithm for a single ϕ-searcher. We show how this algorithm can be extended to handle multiple searchers, and give examples of computed trajectories.

Introduction

In this paper we address a form of the problem known as *pursuit-evasion*. The goal of this problem is to direct the actions of one or more *searchers* through a given environment in such a way as to guarantee that any *evaders* present in the environment will be found. As an example, consider the problem of closing a museum for the night. In order to be sure that no thieves or other malcontents remain inside after closing, the guards must perform a thorough search of the building. They must keep in mind that any intruders may try to avoid the guards. For example, if a guard is checking each room along a hall, an intruder might sneak behind the guard while he is checking one room and hide in a room that was already checked. In this case, one solution might be to use two guards, with one always keeping watch on the hall.

Our goal is to derive strategies for robots that allow them to play the role of guard. In particular, we are interested in scalable techniques for coordinating the actions of teams of robots to clear entire buildings. In this paper, we establish an analytical foundation for studying this problem by introducing the concept of a ϕ-searcher, which is a robot equipped with a ϕ-radian field of view (FOV) sensor for detecting evaders. The ϕ-searcher reflects the realities of physical robots, most of which have limited FOV sensors, and thus the techniques we develop can be applied to robots. Furthermore, the ϕ-searcher is a qualitatively different kind of searcher from those previously studied in pursuit-evasion scenarios, and so warrants the fresh analysis that we present here.

After proving that computing the minimum number of ϕ-searchers required to search an environment is NP-hard, we present the first complete search algorithm for the case of one ϕ-searcher in a known environment. We show how this algorithm can be extended to handle multiple robots (albeit at a loss of completeness). We have implemented and tested this algorithm in a variety of environments and present example solution trajectories.

Background and related work

The first rigorous formulation of the pursuit-evasion problem is due to Parsons (1976), who restricted his study to the case in which the environment is a discrete graph. Nothing is known about the location or motion of the evader, who is assumed to be able to move arbitrarily fast through the graph. The evader can occupy any edge in the graph; to find the evader, a searcher must walk along the edge occupied by the evader and "touch" the evader. The entire graph is initially *contaminated*, which means that the evader could be anywhere. As the search progresses, an edge is *cleared* when it is no longer possible for the evader to occupy that edge. Should it later happen that the evader could have moved back to a previously clear edge, that edge is said to be *recontaminated*. Using this terminology, the goal of the problem can be restated as follows: find a trajectory for each searcher such that the entire graph is cleared.

A visibility-based version of the pursuit-evasion problem was introduced by Suzuki & Yamashita (1992), who changed the domain from discrete graphs to continuous polygonal free spaces and coined the term k-*searcher*. In this formulation, in order to find an evader, a k-searcher need not touch the evader, but can instead "see" the evader from a distance. The k-searcher is equipped with k infinitely thin "flashlights" with which it can search the environment. These flashlights have unlimited range (but cannot see through walls) and can be freely rotated about the searcher at bounded speed and independently of the searcher's motion. Commonly studied are the cases when $k = 1$, $k = 2$, and $k = \infty$ (LaValle *et al.* 1997; Guibas *et al.* 1999; Lee, Park, & Chwa 2002). The ∞-searcher can see in all directions at once.

In addition to such analytical study, there has recently been some work on forms of pursuit-evasion with physical robots. Roy & Gordon (2002) model the single-robot action-selection problem as a POMDP, which is made tractable by compression of the sparse belief space. A similar probabilistic framework is employed by Vidal *et al.* (2002), who use heuristic search to find strategies for coordinating teams of air and ground vehicles to search an unknown outdoor environment. More distantly related is the large body of work on cooperative tracking of moving targets with fixed sensors and/or mobile robots (Werger & Matarić 2001; Stroupe 2003; Spletzer & Taylor 2003).

The ϕ-searcher

The existing body of analytical work on visibility-based pursuit-evasion is concerned with some form of the k-searcher, and is not readily applicable to physical robots, few of which are equipped with movable flashlights or omnidirectional sensing. Since our target domain is teams of physical robots, we introduce a new class of searcher, which to our knowledge has not yet been studied, and which we call the *ϕ-searcher*. The *ϕ-searcher* is a holonomic (i.e., omnidirectional drive) mobile robot with a limited FOV sensor having angular aperture $\phi \in (0, 2\pi]$. The sensor has unlimited range, but cannot penetrate obstacles. This robot can move (i.e., rotate and/or translate) at bounded speed. When $\phi = 2\pi$, we have an ∞-searcher. For $\phi < 2\pi$, however, we have a different kind of searcher, with significantly diminished capabilities. Since the sensor's FOV can be freely rotated about the searcher at bounded speed and independently of the searcher's motion (this follows from the holomonicity of the robot), the capabilities of the ϕ-searcher lie somewhere between those of a 1-searcher and those of a 2-searcher. Shown in Figure 1 is an example of a ϕ-searcher, for $\phi = \pi$.

Given a connected polygonal free space F, the pursuit-evasion problem is to find a trajectory through F (called a *search schedule*) for ϕ-searchers that guarantees detection of an arbitrarily fast *evader*, whose trajectory and initial location are unknown.[1] Analogously to the graph search problem, any part of F where the evader can be hiding is called *contaminated* and any part of F where the evader cannot be hiding is called *clear*. Whenever there exists a path between contaminated space and clear space, that clear space is said to be *recontaminated*. The space F is initially contaminated and the goal is to clear it.

It is known that for the discrete graph search problem, establishing the minimum number of searchers required to search a given graph, known as the *search number* of the graph, is NP-hard (Megiddo *et al.* 1988). For the visibility-based version, establishing the minimum number of ∞-searchers required to search a given polygonal free space is also NP-hard (Guibas *et al.* 1999). We have a similar result for ϕ-searchers:

Theorem 1. *Computing the minimum number of ϕ-searchers required to search a given polygonal free space F is NP-hard.*

Proof. We use the same reduction as Guibas *et al.* (1999): any planar graph G can be mapped onto an equivalent polygonal free space F by making vertices into convex rooms and edges into "kinked" hallways. Each hallway has a kink, or bend, in the middle, such that the searcher cannot see from one end of the hall to the other. As a result, to clear a hall, the searcher must walk its entire length, just as with an edge in G. The kink removes all advantages of visibility, and thus has the same effect on the ϕ-searcher as it does on the ∞-searcher. Since any planar graph can be mapped onto an equivalent polygonal free space, and since computing the search number of a planar graph with maximum vertex degree 3 is NP-hard (Monien & Sudborough 1988), computing the minimum number of ϕ-searchers required to search a polygonal free space is also NP-hard. □

A complete algorithm for a single ϕ-searcher

Since, by Theorem 1, we cannot easily determine the minimum number of ϕ-searchers required to search a given environment, we focus initially on the case of controlling a single ϕ-searcher. In this section, we present a complete search algorithm for the case of a single ϕ-searcher. That is, we are interested in finding a trajectory for a single ϕ-searcher that will search a given polygonal free space F, under the assumption that such a trajectory exists. We address the extension to multiple searchers later.

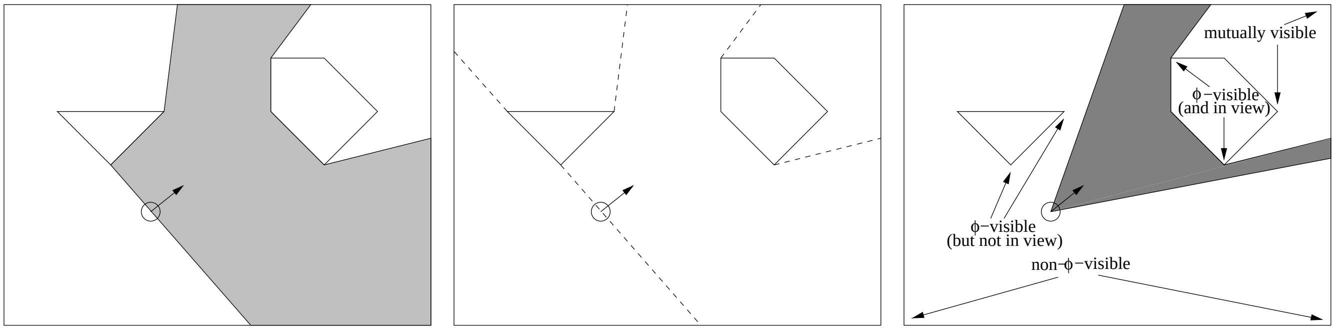
We take inspiration in the design of our algorithm from the work of Guibas *et al.* (1999), who have previously given a complete algorithm for the case of a single ∞-searcher. Their algorithm does not suffice for a ϕ-searcher with $\phi < 2\pi$, because it does not account for the searcher's limited FOV. However, we borrow from their work in several ways.

The basic steps of our algorithm are: (i) by a series of partitions, retract the given free space into a network of curves that represent the visibility constraints induced by the environment and the searcher's FOV; (ii) construct an *information graph* that encodes the possible information states of the problem as the searcher moves, using the network of intersecting curves as a roadmap; then (iii) search this graph for a goal state, and read the desired trajectory out from the resulting path. The key to this algorithm is identifying the critical points in the environment; it is only by moving through these points that the searcher will change the information state of the problem.

Identifying critical changes in information state

The area visible to a ϕ-searcher when it is positioned at a point p with orientation θ in F is called its *visibility polygon*, $V_\phi(F, p, \theta)$, abbreviated to V_ϕ when F, p, and θ are clear from the context. When a point q lies within V_ϕ, we say that q is *in view* (Figure 1(c)). The visibility polygon is

[1] It may be the case with physical robots that the available map, having been acquired from sensor data, is grid-based, rather than polygonal. If so, then the first step is to generate an approximate polygonal representation of the grid-based map, either automatically (e.g., via smoothing and line-fitting) or manually (e.g., with the aid of an architectural floorplan).



(a) (b) (c)

Figure 1: *Introduction to the ϕ-searcher Shown in (a) is an example of a π-searcher positioned at a point p with orientation θ in a polygonal free space F. The shaded region is the searcher's visibility polygon, $V_\pi(F, p, \theta)$. Shown as dashed lines in (b) are the induced gap edges for the π-searcher, which partition the free space into 4 regions. Shown in (c) are examples of vertices that are* in view, visible, *and/or* ϕ-visible, *for $\phi = \frac{\pi}{3}$.*

defined by a set of line segments, some of which lie on the boundary of F and some of which do not. The latter segments, which cross through the interior of F, are called *gap edges* (Figure 1(b)). These edges, combined with the edges of F, form a *planar map* that partitions the free space of F into a set of *faces*, which we call *regions*. We can attach to each region a binary label that indicates whether it is clear ("0") or contaminated ("1").

We model the information state of the problem as $(p, \theta, B(p, \theta))$, where (p, θ) is the searcher's pose and $B(p, \theta)$ is the list of binary labels on the regions induced by its gap edges. To monitor progress, we need only take notice when this information state changes *combinatorially*; that is, when the set of regions or region labels changes. A combinatorial, or critical, change in information state corresponds to a move of the searcher that changes the topology of contaminated space. During such a move, one or more gap edges will: disappear, split, or merge (it can also happen that a new gap edge appears during a move, but new gap edges always border clear space, and so such a move does not change the topology of the contaminated space).

Other moves can deform the contaminated space of F, but do not change its topology. We can characterize critical changes in information state with the following necessary (though not sufficient) condition:

Lemma 1. *Given a single ϕ-searcher in a polygonal free space F, there can be a change in the topology of the contaminated space in F only if there is a change in the set of vertices of F that lie in V_ϕ.*

Proof. We treat the three cases separately:

1. **Edge disappearance.** For a gap edge e to disappear, one of two events must occur. Either the concave vertex of F that induced e moves out of V_ϕ, or e becomes coincident with the boundary of F and terminates at a previously non-visible convex vertex of F. In the first case, e disappears because it falls out of view (and thus is no longer part of V_ϕ); in the second case, e disappears because it becomes part of the boundary of F (and thus is no longer a gap edge).
2. **Edge splitting.** A gap edge e can be split only if a previously non-visible concave vertex of F, say v, moves into V_ϕ such that v lies on e. The gap edge e will then split into two new edges, with the first terminating at v and the second originating at v.
3. **Edge merging.** Merging is simply the reverse of splitting. Two gap edges $e1$ and $e2$ can merge only if $e1$ terminates at the same concave vertex of F, say v, from which $e2$ originates, and v moves out of V_ϕ. The result is a single new edge, having the same origin as $e1$.

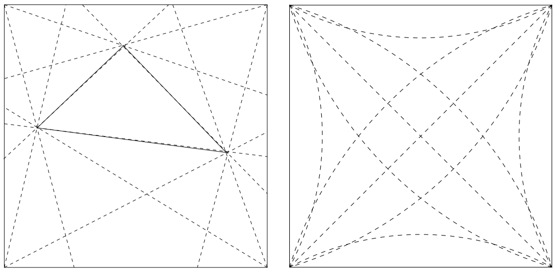
In each case, at least one vertex of F moves into or out of V_ϕ. □

This lemma tells us that any critical change in information state will be accompanied by (actually, caused by) a change in the set of vertices of F that lie within V_ϕ. So we need to identify the points at which there can be a change in the set of vertices that are in view.

Before continuing, we define two notions of visibility from a point $p \in F$ (Figure 1(c)). We say that two points p and q are *mutually visible* if the line segment between p and q does not cross the boundary of F (this is the traditional geometric notion of visibility). Equivalently, we may say that p is *visible* from q or that q is *visible* from p. We say that a pair of points (p, q) is ϕ-*visible* from a point s if there exists an orientation, say θ, for a ϕ-searcher located at s such that both p and q lie within V_ϕ, with p and q ordered counterclockwise about s. That is, if we rotate a sweep line counterclockwise about s through the FOV of a searcher positioned at s with orientation θ, we encounter both p and q, with p coming before q (angular ties are broken by distance from s). Clearly, if (p, q) is ϕ_1-visible from s, then (p, q) is also ϕ_2-visible from s for any $\phi_2 \geq \phi_1$. Thus, if the pair (p, q) is ϕ-visible from s, then p and q are also both visible from s (traditional visibility is just 2π-visibility).

Partitioning the environment

We now partition F into a set of regions R. Our goal is that the following condition hold for each region $r \in R$: given



(a) (b)

Figure 2: *Shown in (a) is the partition $P_\pi(F)$ for a space F consisting of a square containing a triangular hole. Shown in (b) is the partition $P_\phi(F)$ for a square free space F, with $\phi = \frac{2\pi}{3}$. In both cases, the dashed lines are the segments used in the construction of the partition (the boundaries of the square and/or triangle are also included). Either visibility (a) or ϕ-visibility (b) of boundary vertices is constant across the resulting regions.*

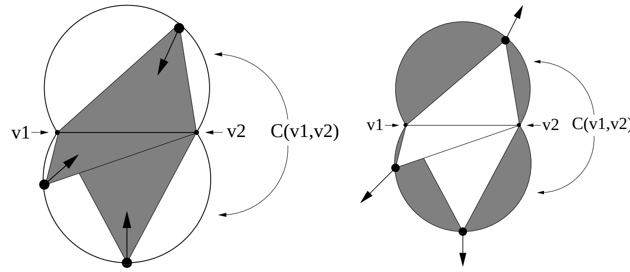
a ϕ-searcher located anywhere in r, the set of ϕ-visible vertex pairs does not change. First, we ensure that there is no change in the set of visible vertices. For each pair of mutually visible vertices v_1 and v_2 of F (including when v_1 and v_2 are endpoints of the same segment of F), we construct the segment v_1v_2, then extend this segment in either direction as far as possible without crossing the boundary of F. The intersections of the resulting segments form a partition of F into a set of convex regions. The intuition behind this technique is that we identify all places where the set of visible vertices can change due to occlusion (Figure 2(a)). The resulting partition, $P_\pi(F)$, has the following property:

Lemma 2. *Given any region r from the partition $P_\pi(F)$ of a polygonal free space F, the set of visible vertices of F is the same for all points in the interior of r.*

Proof. Assume by contradiction that some vertex v_1 of F is visible from a point p and not visible from another point q, with both p and q in the interior of r. Consider any continuous path γ from p to q such that γ is contained in the interior of r. Then there is some point along γ, say s (possibly equal to q), where v_1 disappears from view. For this to occur, there must be another vertex of F, say v_2, that occludes v_1. The three points v_1, v_2 and s will be collinear; v_1 and v_2 will be mutually visible; and v_2 and s will be mutually visible. Then in the construction of P_π, the extension of the segment v_1v_2 would pass through s, which means that s does not lie in the interior of r. Thus there is no path between p and q that remains in the interior of r, which contradicts the assumption that both p and q lie in the interior of r. $\square$

So we know that the set of visible vertices cannot change within a region $r \in P_\pi(F)$. However, for $\phi < 2\pi$, it is still possible for the set of ϕ-visible vertex pairs to change within a region. We want to refine the partition $P_\pi(F)$ so that there is no change in ϕ-visibility of vertices as a ϕ-searcher moves within a single region. For this purpose, we introduce *visibility curves*:

Definition (Visibility curve). *Given $0 < \phi \leq \pi$ and two distinct points v_1 and v_2, the* visibility curve, *denoted $C_\phi(v_1, v_2)$, is the locus of ϕ-searcher positions closest to the midpoint of the segment v_1v_2 such that an appropriate orientation of the searcher will place both v_1 and v_2 in V_ϕ. For $\pi < \phi < 2\pi$, $C_\phi = C_{(2\pi-\phi)}$. The curve C_ϕ is undefined for $\phi = 2\pi$.*



(a) (b)

Figure 3: *The visibility curve $C_\phi(v_1, v_2)$ for two points v_1 and v_2. Shown in (a) is the case when $\phi = \frac{\pi}{3}$; these arcs are the closest that a $\frac{\pi}{3}$-searcher can come to the midpoint of v_1v_2, while maintaining visibility of both points. Shown in (b) is the case when $\phi = \frac{5\pi}{3}$; these arcs are the farthest that a $\frac{5\pi}{3}$-searcher can move after crossing v_1v_2 while maintaining visibility of both points. Three example poses along each curve are shown, with the relevant portions of the visibility polygons shaded.*

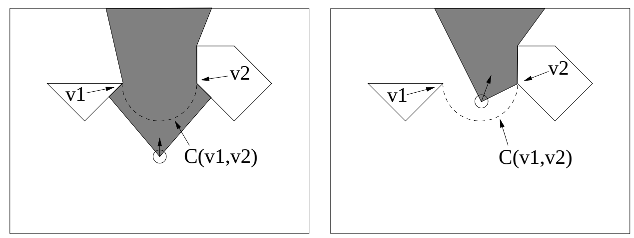


Figure 4: *An example of how ϕ-visibility changes when crossing a visibility curve, for $\phi = \frac{\pi}{2}$. As the searcher crosses $C_{\frac{\pi}{2}}(v_1, v_2)$, it must lose sight of at least one of v_1 and v_2.*

As can be seen in Figure 3(a), the visibility curve for $0 < \phi < \pi$ is composed of two arcs. Each arc is part of a circle defined by the locus of points from which the segment v_1v_2 subtends the angle ϕ (the segment v_1v_2 is always a chord in each circle; when $\phi = \frac{\pi}{2}$ the two circles are the same and the segment v_1v_2 is a diameter). To maintain visibility of both vertices from a point along this curve, the searcher must be oriented toward the midpoint of v_1v_2. As ϕ approaches π, these arcs flatten, until they meet to form a single line when $\phi = \pi$. In this case, a π-searcher positioned on the line must be oriented orthogonal to the line in order to maintain visibility of v_1 and v_2. As shown in Figure 3(b), for $\pi < \phi < 2\pi$, the visibility curve is composed of the same two arcs as for $2\pi - \phi$, but the orientation of the searcher is rotated by π.

The importance of the visibility curve is that it makes concrete the constraints induced by the requirement to maintain visibility of a given pair of vertices with a ϕ-searcher. For $0 < \phi \leq \pi$, the visibility curve is the closest that a ϕ-searcher can approach two vertices while keeping them both in view (Figure 4). If the searcher moves forward from a point on this curve, it will necessarily lose visibility of at least one vertex. For $\pi \leq \phi < 2\pi$, the interpretation is slightly different: if two vertices are currently in view and a ϕ-searcher passes between them, the visibility curve is the farthest that the searcher can travel while keeping both vertices in view. When the searcher reaches a point on the visibility curve, its blind spot becomes wedged between the two vertices; any further forward motion will cause at least one vertex to fall out of view.

We now refine our partition of F with a set of visibility curves. For every pair of vertices v_1 and v_2 of F (including when v_1 and v_2 are endpoints of the same segment of F) for which there exists some point on the curve $C_\phi(v_1, v_2)$ that lies in the free space of F and from which (v_1, v_2) and/or (v_2, v_1) is ϕ-visible , we add $C_\phi(v_1, v_2)$ to F. The intuition behind this step is that we need only include those curves that represent visibility changes that can actually occur for a searcher moving through F. For example, if the searcher can never simultaneously see two vertices, then the visibility curve between them is not a meaningful constraint on the searcher's motion. For the same reason, we discard from each curve any portion that lies outside F. It is possible to further prune the visibility curves by removing from $C_\phi(v_1, v_2)$ *any* portion from which one or both of v_1 and v_2 is not visible. This further pruning would speed up trajectory planning, but has no effect on the completeness of our algorithm.

Note that when $\phi = \pi$, the addition of visibility curves is redundant: the existence of a third point on $C_\pi(v_1, v_2)$ from which v_1 and v_2 are ϕ-visible is equivalent to v_1 and v_2 being mutual visible, and $C_\pi(v_1, v_2)$ is just the line through v_1 and v_2, which was included in the coarser partition (hence the name $P_\pi(F)$).

The intersections of the resulting arrangement of curves and lines form a partition of F into a set of (not necessarily convex) regions (Figure 2(b)). This partition, $P_\phi(F)$, has the following two properties:

Lemma 3. *Given any region r from the partition $P_\phi(F)$ of a polygonal free space F, the set of ϕ-visible vertices is the same for all points in the interior of r.*

Proof. Since $P_\phi(F)$ is a refinement of $P_\pi(F)$, r must be contained within a single region, say r_π, of $P_\pi(F)$. As the set of visible vertices cannot change within r_π (Lemma 2), the set of visible vertices likewise cannot change within r.

So we are left to show that, among the (constant) set of visible vertices, the set of ϕ-visible vertices does not change. Assume by contradiction that this condition does not hold. In particular, assume that the vertex pair (v_1, v_2) is ϕ-visible from some point p and not ϕ-visible from some other point q, with both p and q in the interior of r. Since (v_1, v_2) is ϕ-visible from p, both v_1 and v_2 must also be visible from p. And since the set of visible vertices is the same for both p and q (because they are both in r), v_1 and v_2 must also be visible from q. Then, by the definition of a visibility curve, p and q must lie on opposite sides of $C_\phi(v_1, v_2)$, which contradicts the assumption that p and q lie in the same region r. □

Lemma 4. *Given any region r from the partition $P_\phi(F)$ of a polygonal free space F, any vertex pair that is ϕ-visible from the interior of r is also ϕ-visible from the boundary of r.*

Proof. Assume by contradiction that the vertex pair (v_1, v_2) is ϕ-visible from some point p in the interior of r (by Lemma 3, the particular choice of p makes no difference), and not ϕ-visible from some point q on the boundary of r. It is clear that, since v_1 and v_2 are visible from p, they must also be visible from q. Then, by the definition of a visibility curve, p and q must lie on opposite sides of $C_\phi(v_1, v_2)$, which contradicts the assumption that q is on the boundary of the region containing p. □

These lemmas tell us that a ϕ-searcher that is positioned within a region r can move onto the boundary of r without losing visibility of any vertices (it may gain visibility of some vertices). At this point, to avoid possible ambiguity, it will be necessary to order to the set of vertices that are in view.[2] For a ϕ-searcher positioned at p with orientation θ, we denote by $S_\phi(F, p, \theta)$ the ordered set of vertices of F that are in view (i.e., in $V_\phi(F, p, \theta)$), sorted counterclockwise about the searcher (angular ties are broken by distance from the searcher). For brevity we may abbreviate this set as S_ϕ. We now show that a searcher positioned in the interior of a region can move to the region's boundary without removing or reordering any vertices that are initially in view:

Lemma 5. *Given any region r from the partition $P_\phi(F)$ of a polygonal free space F, and a ϕ-searcher with pose (p, θ) such that p lies in the interior of r, the searcher can move to any point q on the boundary of r without removing or reordering any vertices in S_ϕ.*

Proof. Consider any continuous path γ from p to q such that $\gamma \setminus q$ is contained in the interior of r and q is visited exactly once (i.e., the path γ intersects the boundary of r only at q, where it terminates). Each vertex pair that is in $S_\phi(F, p, \theta)$ must be ϕ-visible from p and so, by Lemmas 3 & 4, also be ϕ-visible from all other points on γ (since γ is confined to r). So it is possible for the searcher to move continuously along γ without losing sight of any vertices. Furthermore, because ϕ-visibility guarantees an ordering on each vertex pair, it is possible for the searcher to move continuously along γ such that no vertex pair in S_ϕ is reordered. As a result, the searcher can move from p to q without removing or reordering any vertices in S_ϕ. □

This lemma tells us that we can move a searcher from the interior of a region to its boundary, with the only change in S_ϕ being the *addition* of vertices of F. To put it another way, a searcher that is positioned in the interior of a region may as well move to the region's boundary, for at the worst its vertex coverage will remain the same. As a result, without loss

[2] When $\phi > \pi$, it may be possible to change the information state without changing the set of vertices that are view, by wedging the searcher's blind spot between different vertex pairs. In this case, only the *order* of the in-view vertices will change, and so we must keep track of this order to be able to distinguish among these information states.

of generality we can restrict our attention to the boundaries of the regions of $P_\phi(F)$, and ignore their interiors. Furthermore, no vertices S_ϕ will be removed or reordered as the searcher moves along a single component (i.e., segment or arc) of the boundary of a region; such a change can only occur at the intersection of two components. By moving to an intersection, the searcher is in fact moving from the interior to the boundary of a larger region that would exist if the boundary component along which it is traveling were excluded from the partition. So we can further focus our attention on the intersections of region boundaries.

Of course, the searcher may have to rotate as it moves; ϕ-visibility only guarantees that it is *possible* for the searcher to simultaneously see a given pair of vertices. In general, for a point p that is an intersection of one or more region boundaries, there will be an interval of orientation for which the set of vertices in V_ϕ will remain unchanged. Equivalently, there is an interval of orientation for which each ϕ-visible vertex pair is in V_ϕ. By intersecting all such intervals, we can partition the space of possible searcher orientations, $[0, 2\pi)$. The intervals produced by this partition, $\Theta_\phi(F, p)$, have the following property: the set of vertices in V_ϕ is the same for any two orientations in the same interval.

So all orientations in a given interval, say i, of $\Theta_\phi(F, p)$ are equivalent, in the sense that there can be no critical change in information as a ϕ-searcher, positioned at p, rotates within i. We can identify each orientation interval by a pair of vertices (v_{first}, v_{last}), which are, respectively, the first and last vertices in view, according to the order in which each vertex is encountered by rotating a sweep line counterclockwise through the searcher's FOV. Without loss of generality, we can assume that a ϕ-searcher that is oriented within an interval (v_{first}, v_{last}) is always oriented such that either its minimum FOV boundary intersects v_{first} or its maximum FOV boundary intersects v_{last}.

Building the information graph

We now have the roadmap along which the searcher will travel as it moves through F, and we can build a directed *information graph*, G_I, that encodes all possible critical changes in the information state. For each point p that is the intersection of two or more components of region boundaries in $P_\phi(F)$, compute the orientation intervals $\Theta_\phi(F, p)$. We add to G_I a node for each possible region labeling that can result from placing the searcher at p and rotating it through each orientation interval in $\Theta_\phi(F, p)$.

We complete the construction of G_I by adding edges that correspond to the feasible changes in information state. First we must define the actions that are allowed for the searcher. Assume the searcher is positioned at a point p that is the intersection of two or more region boundary components, and oriented within an interval defined by (v_{first}, v_{last}). This searcher can:

1. Rotate into any adjacent orientation interval.
2. Translate along the length of any boundary component c incident to p such that the orientation interval (v_{first}, v_{last}) is valid at the opposite end of c. During the move the searcher servos its rotation so as to maintain visibility of both v_{first} and v_{last} (and thus all vertices in between).

For each node n in G_I, we determine the information states that would result from taking each allowable action at the configuration that n represents, and add edges from n to the nodes that encode these new states.

Given an information state from which to start, we can then search G_I for a goal state, which represents all of F being clear. In a start state, all in-view regions are clear and all others are contaminated. In a goal state, all regions are clear. Given a feasible path through G_I from start to goal, we can read from it the required searcher trajectory, as a series of in-place rotations and constrained translations. We now establish the completeness of this approach:

Theorem 2. *An algorithm that will find any path to a goal vertex from a start vertex in G_I is complete for the visibility-based pursuit-evasion problem with a single ϕ-searcher.*

Proof. Given a polygonal free space F and a single ϕ-searcher, take any solution trajectory, say τ. Our goal is to map τ onto an equivalent path in G_I. That is, we wish to show that τ can be transformed into an equivalent trajectory along the curves that make up the roadmap used in constructing G_I.

Denote by (p, θ) the searcher's pose at the start of τ. If p lies in the interior of a region r in $P_\phi(F)$ then, by Lemma 5, we can move the searcher to any point on the boundary of r, without losing or reordering any in-view vertices of F. In particular, we can move the searcher to any intersection along the boundary of r. If p already lies on a region boundary, then we can slide the searcher along that boundary to the nearest intersection, again without losing or reordering any in-view vertices.

As τ progresses, we can replicate its moves within our roadmap in the following way. While τ remains in the interior of a region r, we keep the searcher at an intersection, say q, on the boundary of r, and achieve any necessary changes in vertex visibility by simply rotating the searcher through its orientation intervals. If τ crosses into an adjacent region r' such that q is also on the boundary of r', we can continue to rotate the searcher in place. If, on the other hand, q is not on the boundary of r', we move the searcher along any boundary component that is incident to both q and some other intersection, say s, that does lie on the boundary of r'. During this translation, the searcher's rotation is controlled so as to remain within its current orientation interval and not lose or reorder any in-view vertices (Lemma 5 guarantees that this is possible).

In this way, we create a new trajectory τ' that moves along the roadmap and is at as least as good as τ. That is, S_ϕ at each point on τ' is a superset of S_ϕ at the corresponding point on τ. Compared to τ, a searcher moving along τ' may see additional vertices, but it will never miss any, nor will it see them in a different order. As we can construct a solution τ' given any solution τ, the algorithm is complete. □

Multiple searchers

One way to extend this algorithm to handle multiple searchers is to include in the construction of G_I the joint information and action spaces of all searchers. Unfortunately, this approach is not complete. The reason is that the visibility polygons of multiple searchers can interact and overlap in such a way that the information state can change without any searcher crossing a region boundary in P_ϕ.

A more serious drawback to extending our algorithm to multiple searchers in this way is that the joint information and actions space grow exponentially in the number of searchers. Even in simple environments and with only two searchers, the information graph G_I requires a prohibitively large amount of memory to store and a correspondingly long time to search. Nevertheless, we have implemented this extension, and show in the next section that it computes valid clearing trajectories.

Implementation and computed examples

We have implemented and tested the algorithm described above, using the CGAL computational geometry library (Burnikel *et al.* 1999). For reasons of efficiency and convenience, our implementation thus far computes trajectories only for the special case of $\phi = \pi$. In this case, the roadmap in $P_\phi(F)$ consists only of linear objects, and so the underlying geometric computation can be done with rational numbers. For $\phi \neq \pi$, circular arcs are introduced, and the exact computation of their intersections requires the use of arbitrary precision real numbers, which are far slower to work with than rationals. Efficiency concerns aside, the case of $\phi = \pi$ is of particular interest and value to roboticists, because a sensor that is commonly found on robots today is a scanning laser range-finder with a $180°$ FOV (e.g., the SICK LMS). So trajectories computed by our implementation can be executed directly on holonomic robots that are equipped with such sensors.

We now present a few computed examples. In all the figures, light gray (or yellow) areas are currently in view (and thus clear), white areas are clear (but not in view), and dark gray areas are contaminated. For reasons of space, not all steps in each trajectory are shown. For clarity of presentation, the searcher's approximate trajectory is shown in the last frame. Shown in Figure 5 is an office-like environment composed of a hallway connecting two rooms. Shown in Figure 7 is a more complex environment, for which the computed trajectory had 42 steps. Figure 6 shows a simple environment that requires two searchers to clear (a single ϕ-searcher is incapable of clearing any loops, even when $\phi = 2\pi$). We used the multi-searcher extension mentioned above to compute this solution, in which two searchers work together to clear the environment.

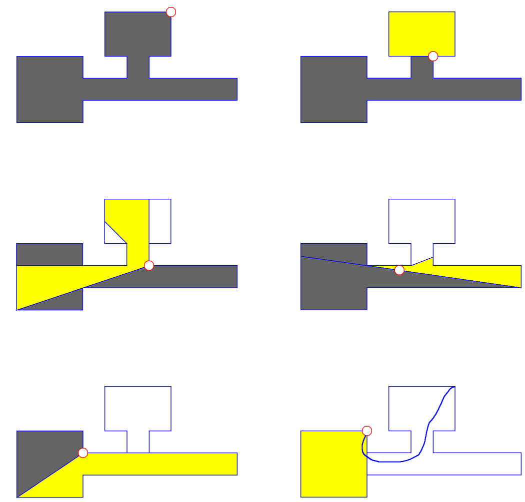


Figure 5: *A computed clearing trajectory for a π-searcher. In this case the searcher clears the environment by moving backward out of the upper room, down the hall, and into the left room.*

Summary

We have presented a novel form of the visibility-based pursuit-evasion problem by introducing a new class of searcher, the *ϕ-searcher*, which we chose because it can readily be instantiated as a physical mobile robot. Because the ϕ-searcher is qualitatively different from the previously studied k-searcher, existing motion-planning techniques do not suffice. As part of a detailed analysis of this new kind of searcher, we showed that computing the minimum number of ϕ-searchers required to search a given environment is NP-hard and derived the first complete search algorithm for a single ϕ-searcher. We showed how this algorithm can be extended to find strategies for multiple searchers by planning in their joint action space, and gave examples of computed trajectories for single searchers and for teams of two searchers.

Planning in the joint action space of all searchers is clearly not the best approach, as it is centralized and scales badly as the number of searchers increases. We are currently designing more parsimonious techniques for coordinating multiple searchers, such as distributed negotiation. For example, if a searcher encounters a hallway with many contaminated rooms, it may ask of another searcher, "Watch my back from the end of the hall while I clear these rooms." It is interesting to note that this kind of cooperative mini-strategy arises naturally in the joint action space plans. We are investigating the possibility of learning from these plans such higher-level primitives as "watch my back" and "guard this intersection."

References

Burnikel, C.; Fleischer, R.; Mehlhorn, K.; and Schirra, S. 1999. Efficient exact geometric computation made easy. In *Proc. of the ACM Symp. on Computational Geometry*, 341–350.

Guibas, L. J.; Latombe, J.-C.; LaValle, S. M.; Lin, D.; and Motwani, R. 1999. A Visibility-Based Pursuit-Evasion Problem. *Intl. J. of Computational Geometry & Applications* 9(4 & 5):471–493.

LaValle, S. M.; Lin, D.; Guibas, L. J.; Latombe, J.-C.; and Motwani, R. 1997. Finding an Unpredictable Target in a Workspace with Obstacles. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 737–742.

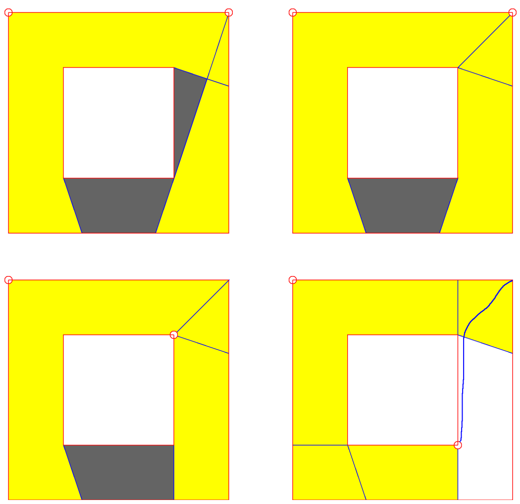


Figure 6: *This example, with one loop, is the simplest environment that requires two π-searchers to clear. In this case, the searcher in the upper left corner remains stationary, watching the upper and left corridors, while the other searcher moves to clear the right and lower corridors.*

Lee, J.-H.; Park, S.-M.; and Chwa, K.-Y. 2002. Simple algorithms for searching a polygon with flashlights. *Information Processing Letters* 81:265–270.

Megiddo, N.; Hakimi, S.; Garey, M.; Johnson, D.; and Papadimitriou, C. 1988. The Complexity of Searching a Graph. *J. of the ACM* 35(1):18–44.

Monien, B., and Sudborough, I. 1988. Min cut is NP-complete for edge weighted trees. *Theoretical Computer Science* 58:209–229.

Parsons, T. 1976. Pursuit-evasion in a graph. In Alavi, Y., and Lick, D., eds., *Theory and Applications of Graphs*, Lecture Notes in Mathematics 642. Berlin: Springer-Verlag. 426–441.

Roy, N., and Gordon, G. 2002. Exponential Family PCA for Belief Compression in POMDPs. In *Proc. of Advances in Neural Information Processing Systems (NIPS)*.

Spletzer, J. R., and Taylor, C. J. 2003. Dynamic sensor planning and control for optimally tracking targets. *The Intl. J. of Robotics Research* 22(1):7–20.

Stroupe, A. 2003. *Collaborative Execution of Exploration and Tracking Using Move Value Estimation for Robot Teams (MVERT)*. Ph.D. Dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Suzuki, I., and Yamashita, M. 1992. Searching for a mobile intruder in a polygonal region. *SIAM J. on Computing* 21(5):863–888.

Vidal, R.; Shakernia, O.; Kim, H. J.; Shim, D. H.; and Sastry, S. 2002. Probabalistic Pursuit-Evasion Games: Theory, Implementation, and Experimental Evaluation. *IEEE Transactions on Robotics and Automation* 18(5):662–669.

Werger, B. B., and Matarić, M. J. 2001. Broadcast of Local Eligibility for Multi-Target Observation. In Parker, L. E.; Bekey, G.; and Barhen, J., eds., *Distributed Autonomous Robotic Systems 4*. New York: Springer-Verlag. 347–356.

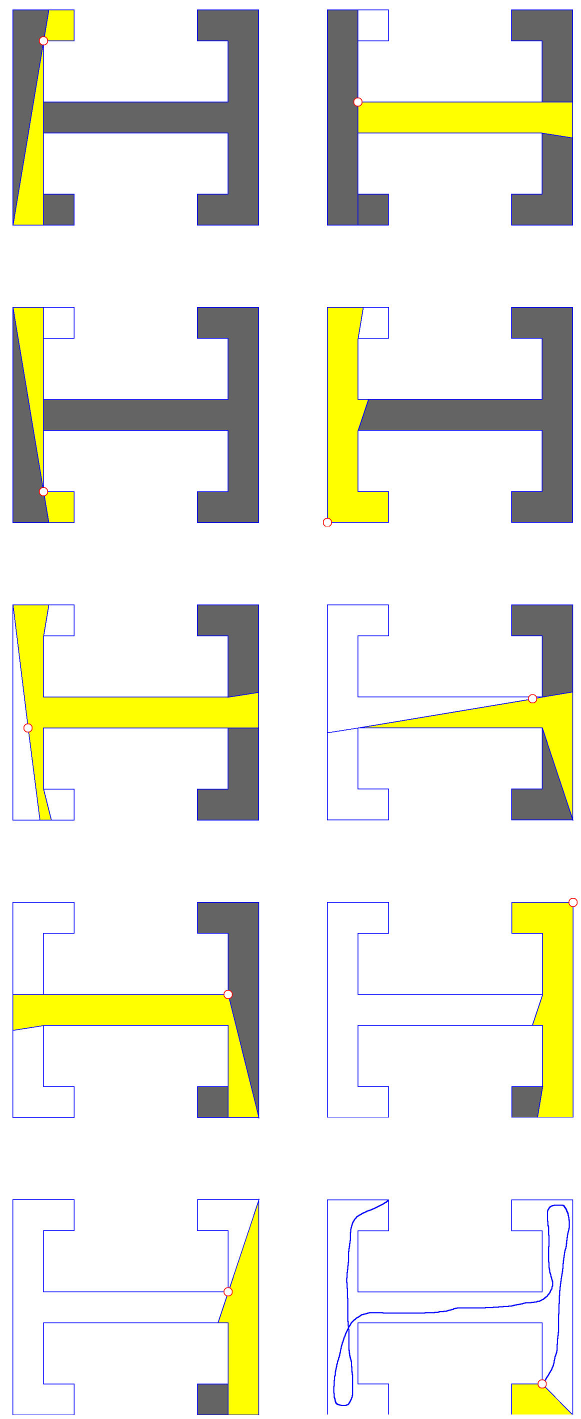


Figure 7: *In this more complicated example, the π-searcher first clears the left hand side from top to bottom, then moves through the horizontal hallway to clear the right hand side, from top to bottom.*