To appear in *Intl. J. of Robotics Research*
April, 2006

Visibility-based pursuit-evasion with limited field of view[*]

Brian P. Gerkey[†] Sebastian Thrun[‡] Geoff Gordon[◇]
gerkey@ai.sri.com **thrun@stanford.edu** **ggordon+@cs.cmu.edu**

† Artificial Intelligence Center, SRI International
‡ Artificial Intelligence Lab, Stanford University
◇ Center for Automated Learning and Discovery, Carnegie Mellon University

Abstract

We study the visibility-based *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 present the first complete search algorithm for a single ϕ-searcher, show how this algorithm can be extended to handle multiple searchers, and give examples of computed trajectories.

1 Introduction

We address the problem known as *pursuit-evasion*. The goal 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 intruders are mobile and 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 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. Our motivation for focusing on sensors with limited FOV is that we wish to derive strategies for robots, and most robots are equipped with such sensors. For example, the popular Pioneer mobile robot is often equipped with a $180°$ array of sonars (Figure 1(a)). Many robots carry more sophisticated sensors, such as laser range-finders (Figure 1(b)) or cameras (Figure 1(c)), but these too tend to have limited FOV.

[*]This paper is an extension of work originally reported in: Gerkey, B. P., Thrun, S. & Gordon, G. (2004), Visibility-based pursuit-evasion with limited field of view, *in* 'Proc. of the Natl. Conf. on Artificial Intelligence (AAAI)', San Jose, California, pp. 20–27.

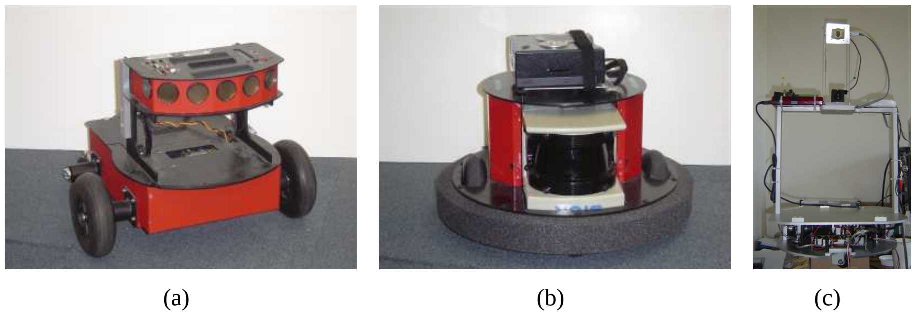

Figure 1: *Some robots equipped with common sensors with limited field of view: (a) sonar array, (b) laser range-finder, and (c) camera.*

So the ϕ-searcher reflects the realities of physical robots and thus the techniques we develop can be applied to robots. Furthermore, the visibility characteristics of the ϕ-searcher are sufficiently different from those of previously studied searchers to warrant the analysis that we present here.

After formally describing the pursuit-evasion problem, we analyze the capabilities of the ϕ-searcher and show that computing the minimum number of ϕ-searchers required to search an environment is NP-hard. We then present the first complete search algorithm for the case of one ϕ-searcher in a known polygonal environment. The key to this algorithm, and the primary contribution of this paper, is the identification of an exact cell decomposition of the searcher's configuration space that reduces the original continuous problem to an equivalent discrete problem. We show how the 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.

2 Background and related work

Pursuit-evasion problems have long been studied from the perspective of differential game theory (Isaacs 1965, Hájek 1975). Given motion models for a pursuer and an evader that move in the open plane, the goal is to determine the conditions necessary for them to collide. Parsons (1976) introduced a rather different formulation (referred to hereafter as *Parsons's problem*) in which the domain is restricted to 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. His motivation was the problem of coordinating a search team to locate a spelunker who has become lost in a network of caves (which can be represented by a graph), but these worst-case assumptions about the evader could equally well describe a more adversarial situation. The spelunker, or 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 Parsons's 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

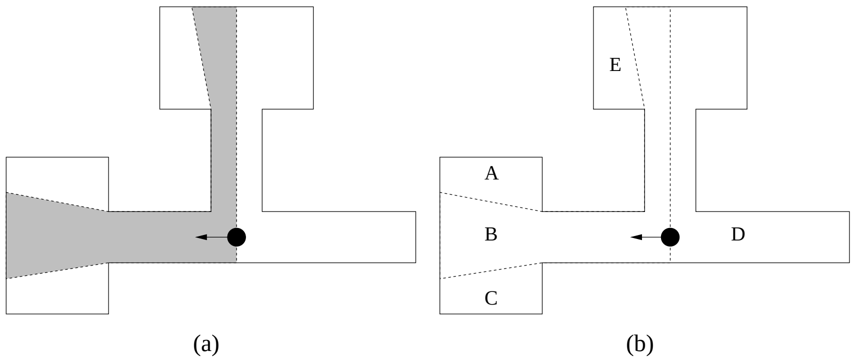

Figure 2: *Introduction to the ϕ-searcher. Shown in (a) is an example of a π-searcher positioned at a point p with an orientation θ in a polygonal free space F. The shaded region is the searcher's visibility polygon, $V_\pi(p, \theta)$. Shown as dashed lines in (b) are the induced gap edges for this π-searcher, which partition the free space into 5 regions (labeled A–E).*

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 et al. 2002). The ∞-searcher can see in all directions at once. A recent interesting result is that any polygonal free space that is searchable by a single ∞-searcher is also searchable by a single 2-searcher (Park et al. 2001), implying some parity of capabilities between the two. Randomized pursuit algorithms have also been studied, in both discrete graphs (Adler et al. 2003) and polygonal free spaces (Isler et al. 2005).

There has also 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. Kalra et al. (2005) use a synthetic market to coordinate the actions of a team of robots executing a "security sweep" of an indoor environment. More distantly related is the large body of work on cooperative tracking of moving targets with fixed sensors and/or mobile robots (Parker 1999, Werger & Matarić 2001, Jung & Sukhatme 2002, Stroupe 2003, Spletzer & Taylor 2003).

As the velocity of the evader approaches zero, the pursuit-evasion problem can be seen as a problem of multi-robot exploration, which has been extensively investigated (Koenig & Simmons 1993, Yamauchi 1998, Burgard et al. 2000). Similarly, as the number of available robots grows large with respect to the size and complexity of the environment, pursuit-evasion can be solved by finding static sensor placements that cover the entire environment. This is known as the Art Gallery problem, and has been studied in great depth (O'Rourke 1987, Shermer 1992).

To our knowledge, limited-FOV pursuit-evasion represents an open problem that has not been previously addressed. Existing analytical work is concerned with some form of the k-searcher, while existing experimental work with robots lacks the rigorous analysis and formal results that we present here. The pursuit-evasion algorithms developed to date are not applicable to the problem that is the topic of this paper.

3 The ϕ-searcher

We introduce a new class of searcher, the *ϕ-searcher*. The *ϕ-searcher* is a holonomic (i.e., omnidirectional drive) mobile robot that moves in the plane and is equipped 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. As $\phi \to 2\pi$, we have an ∞-searcher; as $\phi \to 0$, we have a 1-searcher. For $0 < \phi < 2\pi$, however, we have a different kind of searcher. 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 holonomic capability 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 2 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 for ϕ-searchers that guarantees detection of an *evader* whose trajectory and initial location are unknown.[1] The evader can be arbitrarily small (even a single point) and can move arbitrarily fast, but continuously, through F. 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.

3.1 Capabilities of the ϕ-searcher

We can circumscribe the capabilities of a single ϕ-searcher with the observation that a polygonal free space F is searchable by a single ϕ-searcher only if F is simply-connected. If F contains at least one hole, then an evader can elude a single searcher by moving to keep the hole between them.

A single ϕ-searcher is limited to searching simply-connected environments, but it cannot search all such environments. Given two angles ϕ_1 and ϕ_2, with $\phi_1 > \phi_2$, it is clear that any environment that can be searched by a ϕ_2-searcher can also be searched by a ϕ_1-searcher (e.g., the ϕ_1-searcher can execute the same trajectory as did the ϕ_2-searcher). Less obvious is whether there exists an environment that can be searched by ϕ_1-searcher, but not by a ϕ_2-searcher. That is, does searching capability actually increase with a greater FOV?

We can answer in the affirmative by use of the polygon shown in Figure 3(a), which we call the "jagged E" (inspired by a similar environment due to Suzuki & Yamashita (1992)). Consider the points A, B, and C: any ϕ-searcher pose that provides coverage of A or of C leaves open an uncovered path between the other two points. This is true, regardless of the value of ϕ, even for $\phi = 2\pi$. As a result, it is only possible to search the jagged E in one of two orders: $A \to B \to C$, or $C \to B \to A$. While clearing B, the searcher must not allow an uncovered path between A and C. All other orders (that do not end in one of the given two sequences) would allow earlier work to be undone. For example, if the searcher were to use the order $B \to A \to C$, the point B would necessarily be recontaminated by C while the searcher is clearing A.

Assume without loss of generality that a ϕ-searcher has cleared A and will now clear B. In order to avoid recontamination of A, the searcher must not allow any uncovered path from C to A. In other words, the searcher must simultaneously cover B and the vertical corridor on the left of the jagged E. Such coverage is not possible for all values of ϕ. In fact, the minimum required FOV is equal to the angle labeled

[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).

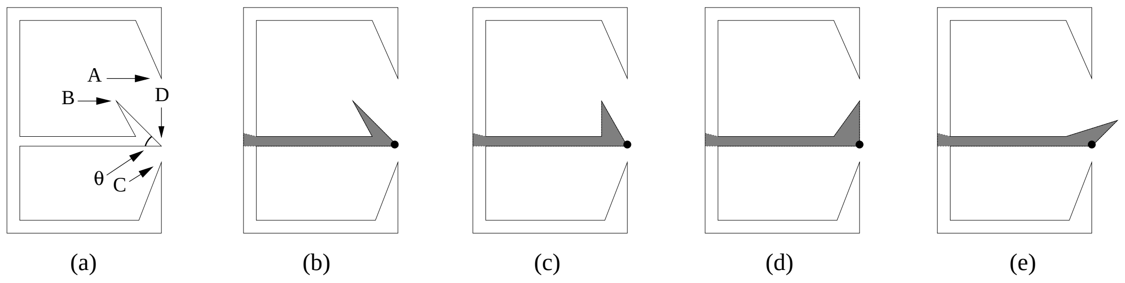

Figure 3: *Shown in (a) is the "jagged E," which demonstrates how ϕ-searcher capability varies continuously with the searcher's field of view. To search this environment, a ϕ-searcher must be able to simultaneously cover the point B and prevent an uncovered path between A and C. The minimum value of ϕ required for this coverage is equal to the angle θ. By varying θ, we can create environments that require a $\frac{\pi}{4}$-searcher (b), a $\frac{\pi}{3}$-searcher (c), a $\frac{\pi}{2}$-searcher (d), a $\frac{3\pi}{4}$-searcher (e), or any other ϕ-searcher, for $0 < \phi < \pi$.*

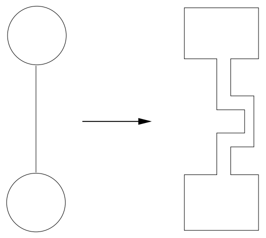

Figure 4: *Mapping a planar graph of nodes and edges onto an equivalent polygonal free space of rooms and hallways. The visibility properties of the original graph are preserved by introducing an occluding kink in each hallway.*

θ in Figure 3(a): a θ-searcher positioned at D can simultaneously cover B and the vertical corridor.[2] For example, Figure 3(b) shows how a $\frac{\pi}{4}$-searcher can accomplish this coverage, when $\theta = \frac{\pi}{4}$.

Now that we know the limiting feature of the jagged E to be the angle θ, we can create environments that are searchable by particular ϕ-searchers. Figures 3(c), 3(d), & 3(e) show the cases when $\theta = \frac{\pi}{3}$, $\theta = \frac{\pi}{2}$, and $\theta = \frac{3\pi}{4}$, respectively. As θ approaches π, so does the required FOV, although when $\theta = \pi$, B becomes visible from the vertical corridor, and so the required FOV is much smaller. In general, we have the following result:

Lemma 1. *Given angles ϕ_1 and ϕ_2, with $0 < \phi_2 < \phi_1 < \pi$, there exists a non-empty set of polygonal free spaces that can be searched by a single ϕ_1-searcher, but not by a single ϕ_2-searcher.*

Proof. Construct a jagged E with $\theta = \phi_1$. The resulting polygon can be searched by a ϕ_1-searcher, but not a ϕ_2-searcher. □

In summary, we have shown that a ϕ-searcher's capability to clear an environment depends on its FOV. Furthermore, we have shown that ϕ-searcher capability varies *continuously* with the value of ϕ. We have also provided a tool, the jagged E, that can be used to construct ϕ-specific environments for any $\phi \in (0, \pi)$.

3.2 Complexity of the general problem

It is known that for Parsons's problem, establishing the minimum number of searchers required to search a given graph, known as the *search number* of the graph, is NP-complete (Megiddo et al. 1988). Guibas et al. (1999) showed that the visibility-based pursuit-evasion problem with 2π-searchers is also intractable, by use of the following equivalence:

Guibas et al.'s (1999) Lemma 1. *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 [with 2π-searchers] on F.*

Figure 4 shows how a graph G can be mapped onto an equivalent polygonal free space F by making nodes 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 2π-searcher, regardless of the value of ϕ. 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-complete (Monien & Sudborough 1988), we can make the following generalization of Guibas et al.'s (1999) complexity result:

Corollary 1. *Given any field of view ϕ and a polygonal free space F, computing the minimum number of ϕ-searchers required to search F is NP-hard.*

4 A complete algorithm for a single ϕ-searcher

Since, by Corollary 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 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.

Guibas et al. (1999) gave a complete algorithm for the case of a single 2π-searcher. Their algorithm does not suffice for a ϕ-searcher with $\phi < 2\pi$, nor can it be easily extended to handle this case, for two reasons. First, the orientation of the searcher must be taken into account, which presents a different configuration space. Second, and more importantly, the searcher's FOV induces new and different visibility constraints that are not captured by Guibas et al.'s (1999) decomposition. However, we borrow from their work in several ways.

We follow an approach known in the robot motion planning literature as *exact cell decomposition* (Schwartz & Sharir 1983, Leven & Sharir 1987, Avnaim et al. 1988, Latombe 1991): decompose the robot's (searcher's) configuration space $\mathcal{C} = \mathbb{R}^2 \times \mathbb{S}^1$ into a set of non-overlapping *non-critical cells* with *critical boundaries*. Intuitively, nothing "critical" can happen as the searcher moves within a cell, whereas "critical" events can occur when the robot crosses a boundary. As a result the searcher can be restricted to trajectories on the cell boundaries. We formalize the meaning of "critical" for the limited-FOV pursuit-evasion problem in the next section.

[2] Actually, the minimum required FOV is slightly smaller than θ, but can be made arbitrarily close to θ by narrowing and extending the middle horizontal corridor that leads to the point B.

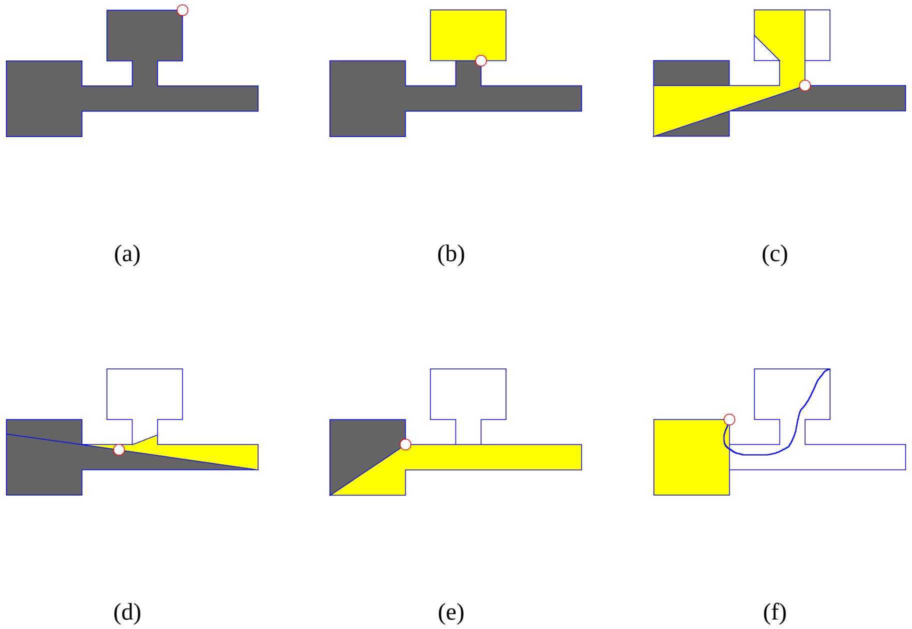
Figure 5: *A computed clearing trajectory for a π-searcher. In this case the searcher clears the environment by moving backward out of the upper room, down the hall, and into the left room (see also Extension 1).*

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, and a primary contribution of this paper, is the identification of the critical configuration space boundaries; it is only by crossing these boundaries that the searcher will change the information state of the problem.

4.1 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*, $V_\phi(p, \theta)$, abbreviated to V when ϕ, p, and θ are clear from the context. When a point q lies within V, we say that q is *in view* (Figure 6). The visibility polygon is 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 2(b)). These edges, combined with the edges of F, form a *planar map* that partitions the free space of F into a set of *regions*.

We can attach to each region a binary label that indicates whether it is clear ("0") or contaminated ("1"). Some who have previously studied pursuit-evasion used the same labeling scheme (Lee et al. 2002), while others labeled the gap edges, rather than the regions (Guibas et al. 1999). 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.

Consider Figure 5, which depicts a computed search trajectory for a single π-searcher in an office-like environment. At each step, the environment is partitioned into finitely many regions by the searcher's visibility polygon. There is at most one region that is currently within the searcher's view, which must be clear, and some other regions that are out of view, each of which may be either clear or contaminated. In the figure, the in-view region is colored light gray, and the out-of-view regions are colored either white (if clear) or dark gray (if contaminated). For example, in Figure 5(c), there are three clear regions (including the one currently in view), and three contaminated regions. So the associated information state might be: $((5,2), \frac{2\pi}{3}, \{1,0,1,1,0,0\})$, assuming that the searcher is positioned at $(5,2)$ with orientation $\frac{2\pi}{3}$, and that the regions are ordered appropriately.

Because they are induced by the searcher's visibility polygon, regions can vary in size and shape as the searcher moves. Regions retain their ordering and contamination labels through such deformations, so the information state does not undergo a critical change. We need only take notice when the information state changes *combinatorially*; that is, when a region is created or destroyed, or when a region's label 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). We can characterize critical changes in information state with the following necessary condition:

Lemma 2. *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.*

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, 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); 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 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. 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. $\square$

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. 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 (Figure 6):

- **mutually visible** : Two points p and q are *mutually visible* iff 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.

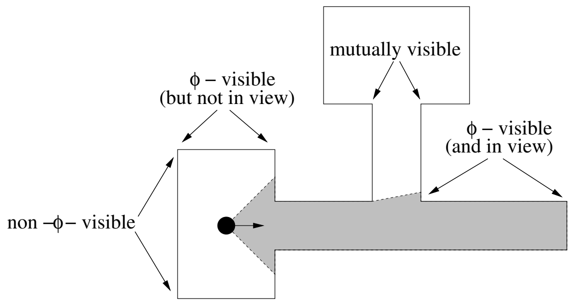

Figure 6: *Examples of vertices that are* in view, visible*, and/or* ϕ-visible*, for* $\phi = \frac{\pi}{3}$.

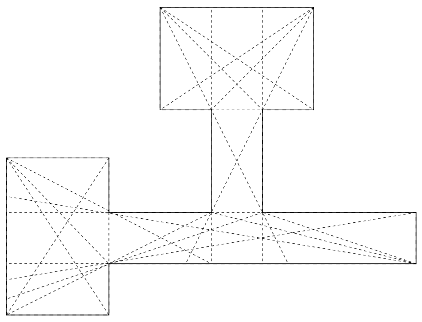

Figure 7: *The partition P_π for a simply-connected polygon. The dashed lines are the segments used in the construction of the partition (the boundaries of the polygon are also included). Vertex visibility is constant within each resulting cell.*

- **ϕ-visible** : A pair of points (p, q) is *ϕ-visible* from a point s iff there exists an orientation, say θ, for a ϕ-searcher located at s such that both p and q lie within $V_\phi(s, \theta)$, 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).

4.2 Decomposing the configuration space

We now decompose the searcher's configuration space $\mathcal{C} = \mathbb{R}^2 \times \mathbb{S}^1$ into cells such that the searcher can move within any cell without changing the set of in-view vertices.[3] We can then restrict our attention to searcher trajectories that remain on the boundaries of these cells. This decomposition, D_ϕ, proceeds in three steps, which are described in subsequent sections:

1. Cast certain lines through F to create the partition P_π; within any cell $r \in P_\pi$, visibility of vertices is invariant.

2. Augment P_π with certain arcs to create the partition P_ϕ; within any cell $r \in P_\phi$, ϕ-visibility of vertices is invariant.

[3]Note that configuration space *cells* are 3-dimensional structures, distinct from the 2-dimensional *regions* that are used to track contamination. Cells are not assigned contamination labels.

3. For each intersection p between lines or arcs in P_ϕ, divide $\mathbb{S}^1$ into orientation intervals to create the partition $\Psi_\phi(p)$; for a ϕ-searcher positioned at p and oriented within any interval $i \in \Psi_\phi(p)$, the set of vertices that are in-view is invariant.

4.2.1 The partition P_π

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 cells. The intuition behind this technique is that we identify all places where the set of visible vertices can change due to occlusion (Figure 7). The resulting partition, P_π, has the following property:

Lemma 3. *Given any cell r from the partition P_π 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. □

4.2.2 The partition P_ϕ

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

Definition (Visibility curve)**.** *Given two points v_1 and v_2 in the plane and a sensor field ϕ, consider the set of points p such that the pair (v_1, v_2) is ϕ-visible from p. This set includes its boundary, which consists of circular arcs that connect v_1 and v_2 and is called the* (ϕ-)visibility curve *of v_1 and v_2, denoted $C_\phi(v_1, v_2)$.*

As can be seen in Figure 8(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 8(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 curve $C_\phi(v_1, v_2)$ is undefined for $\phi = 2\pi$.

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 9). If the

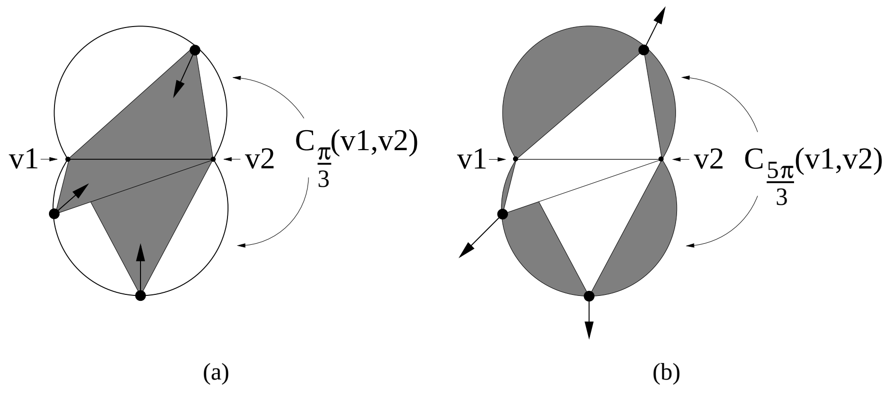

Figure 8: *The visibility curve $C_\phi(v_1, v_2)$ for two points v_1 and v_2. Shown in (a) is the case when $\phi = \frac{\pi}{3}$; these arcs are the closest that a $\frac{\pi}{3}$-searcher can come to the midpoint of v_1v_2, while maintaining visibility of both points. Shown in (b) is the case when $\phi = \frac{5\pi}{3}$; 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.*

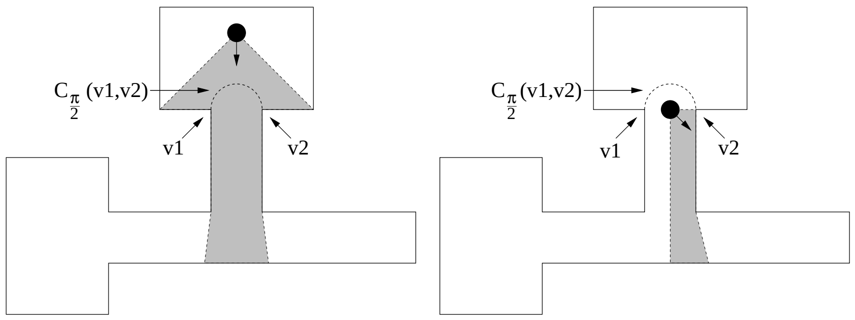

Figure 9: *An example of how ϕ-visibility changes when crossing a visibility curve, for $\phi = \frac{\pi}{2}$. As the searcher crosses $C_{\frac{\pi}{2}}(v_1, v_2)$, it must lose sight of at least one of the two constraining vertices (in this case, v_1).*

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.

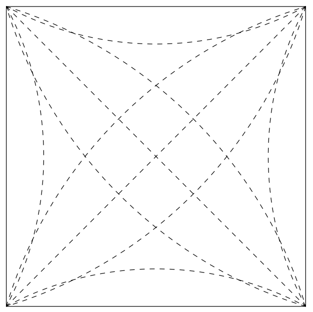

Figure 10: *The partition P_ϕ for a square free space F, with $\phi = \frac{2\pi}{3}$. The dashed lines are the segments and curves used in the construction of the partition (the boundaries of the square are also included). Vertex ϕ-visibility is constant within each resulting cell.*

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 mutually 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_π).

The intersections of the resulting arrangement of curves and lines form a partition of F into a set of cells (Figure 10). This partition, P_ϕ, has the following two properties:

Lemma 4. *Given any cell r from the partition P_ϕ 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_ϕ is a refinement of P_π, r must be contained within a single cell, say r_π, of P_π. As the set of visible vertices cannot change within r_π (Lemma 3), the set of visible vertices likewise cannot change within r.

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. Then p lies inside the set of points from which (v_1, v_2) is ϕ-visible, while q lies outside this set, and so they must lie on opposite sides of the set's boundary. But that boundary, $C_\phi(v_1, v_2)$, is included in the construction of P_ϕ, so p and q must lie in different cells, which contradicts the assumption that they lie in the same cell r. $\square$

Lemma 5. *Given any cell r from the partition P_ϕ 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 4, the particular choice of p makes no difference), and not ϕ-visible from some point q on the boundary of r. Then, by the same reasoning given in the previous proof, p and q must lie on opposite sides of $C_\phi(v_1, v_2)$ (if q were *on* that curve, (v_1, v_2) would still be ϕ-visible), which implies that they lie in different cells, contradicting the initial assumption. $\square$

These lemmas tell us that a ϕ-searcher that is positioned within a cell 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.[4] For a ϕ-searcher positioned at p with orientation θ, we denote by $S_\phi(p, \theta)$ the ordered set of vertices of F that are in view (i.e., in $V_\phi(p, \theta)$), 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 cell can move to the cell's boundary without removing or reordering any vertices that are initially in view:

Lemma 6. *Given any cell r from the partition P_ϕ 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 $S_\phi(p, \theta)$ must be ϕ-visible from p and so, by Lemmas 4 & 5, 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. $\square$

This lemma tells us that we can move a searcher from the interior of a cell 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 cell may as well move to the cell's boundary, for at the worst its vertex coverage will remain the same. As a result, without loss of generality we can restrict our attention to the boundaries of the cells of P_ϕ, 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 cell; 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 cell 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 cell boundaries.

4.2.3 The partition $\Psi_\phi(p)$

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 cell boundaries, there will be an interval of orientation for which the set of vertices in V will remain unchanged. Equivalently, for each ϕ-visible vertex pair (q, s), there is an interval of orientation for which q and s both lie in V. Intersecting any set of such intervals produces a smaller interval; the smallest intervals which we can obtain this way, when taken together, form a partition of $[0, 2pi)$. The intervals produced by this partition, $\Psi_\phi(p)$, have the following property: the set of vertices in V is the same for any two orientations in the same interval.

So all orientations in a given interval, say i, of $\Psi_\phi(p)$ 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 $(v_{\text{first}}, v_{\text{last}})$, 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

[4]When $\phi \geq \pi$, it may be possible to change the information state 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.

through the searcher's FOV. Without loss of generality, we can assume that a ϕ-searcher that is oriented within an interval $(v_{\text{first}}, v_{\text{last}})$ is always oriented such that either its minimum FOV boundary intersects v_{first} or its maximum FOV boundary intersects v_{last}.

It can happen that fewer than two vertices are in view within an interval, in which case v_{first} and/or v_{last} can be $\emptyset$. Then the order of the intervals is used to determine the appropriate orientation. Say, for example, that v_1 and v_2 are the only vertices visible from p, and that (v_1, v_2) is ϕ-visible from p; then $\Psi_\phi(p)$ might be $\{(v_1, v_2), (v_2, \emptyset), (\emptyset, \emptyset), (\emptyset, v_1)\}$. In this case, the third interval, $(\emptyset, \emptyset)$, orients the searcher toward the wall between v_1 and v_2, without either vertex being in view.

4.2.4 The decomposition D_ϕ

Putting it all together, we use the partitions described above to form the decomposition D_ϕ of the searcher's configuration space $\mathcal{C} = \mathbb{R}^2 \times \mathbb{S}^1$. The arcs and lines of P_ϕ decompose $\mathcal{C}$ with respect to the searcher's position. For each point p that is an intersection of two or more lines or arcs in P_ϕ, the intervals of $\Psi_\phi(p)$ further decompose $\mathcal{C}$ with respect to the searcher's orientation. This decomposition has polynomially many cells:

Theorem 1. *Given a polygonal free space F with n vertices, the number of cells in the decomposition D_ϕ is $O(n^5)$.*

Proof. If all pairs of vertices in F are mutually visible (e.g., F is a regular polygon), then the construction of P_π will cast one line from each vertex through each other vertex, resulting in $\frac{n(n-1)}{2}$ lines. Each line can intersect each other line at most once, which gives $O(n^4)$ line/line intersections in P_π.[5]

Similarly, P_ϕ can add at most 2 arcs between each pair of vertices, or $n(n-1)$ arcs in total. Each arc can intersect each other arc at most twice, which gives $O(n^4)$ arc/arc intersections. Each arc can also intersect each line at most twice, which gives $O(n^4)$ arc/line intersections. So we have a total of $O(n^4)$ intersections in P_ϕ.

If a searcher positioned at an intersection p makes one full rotation, each vertex v that is visible from p can enter the searcher's FOV at most once and exit its FOV at most once. Each entry / exit will induce one orientation interval in the partition $\Psi_\phi(p)$. If all n vertices of F are visible from p, then $\Psi_\phi(p)$ can have at most $2n$, or $O(n)$, intervals. Since there are $O(n^4)$ intersections and $O(n)$ intervals per intersection, we have $O(n^5)$ cells in all. □

4.3 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. We add to G_I a node for each possible region labeling that can result from placing the searcher at an intersection p and rotating it through each orientation interval in $\Psi_\phi(p)$.

We add to G_I 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 an intersection p, and oriented within an interval defined by $(v_{\text{first}}, v_{\text{last}})$. This searcher can:

1. Rotate into any adjacent orientation interval.

[5]A tighter bound of $O(n^3)$ is possible here (Guibas et al. 1995), but the difference will be of no consequence in our analysis.

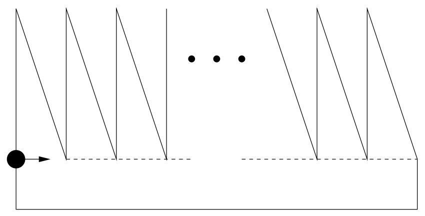

Figure 11: *For a ϕ-searcher with the pose shown here, each "tooth" induces one gap edge, and thus one region in the contamination map. If this polygon has n vertices, then it has $\frac{n-4}{2}$ teeth, which induce $\frac{n-4}{2}$ regions (depending on the value of ϕ, there will be either one or two additional regions). Since each region can take one of two labels, there are $2^{(\frac{n-4}{2})}$ possible labelings, which establishes the tightness of the bound given in Theorem 2.*

2. Translate along the length of any boundary component c incident to p such that the orientation interval $(v_{\text{first}}, v_{\text{last}})$ 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, for each allowable action a, we determine the information state that would result from executing a in configuration n, and add an edge from n to the node that encodes the new state. Given an initial information state and a chosen action, the new information state can be computed as follows: the gap edges are determined by rotational sweep line and the induced planar map is constructed using a traditional sweep line. In the new map, a region r is clear (labeled "0") if either: r is currently in view (because the regions are defined by the searcher's visibility polygon, r will be either entirely in view or entirely out of view), or r does not intersect any contaminated region in the old map, from the previous information state. Otherwise, r is contaminated (labeled "1").

Theorem 2. *Given a polygonal free space F with n vertices, the graph G_I has $O(2^n)$ nodes, and this bound is the best possible.*

Proof. For a ϕ-searcher positioned at an intersection, say p, of P_ϕ and oriented within an interval, say i, of $\Psi_\phi(p)$, each in-view vertex can induce at most one gap edge and thus at most one region in the contamination map. Each region in the map can take one of two labels, and each labeling adds a node to G_I. If all n vertices are in view, then there can be at most n regions and thus $O(2^n)$ labelings for each of the $O(n^5)$ cells in D_ϕ, or $O(2^n)$ nodes in G_I. Figure 11 shows a polygon and ϕ-searcher pose that induce (at least) $2^{(\frac{n-4}{2})}$ region labelings, for any n and any ϕ, establishing the tightness of the bound. $\square$

If we can bound by k the maximum number of vertices that are visible from any point in F, then there will be $O(2^k)$ labelings for each of the $O(n^5)$ cells, and G_I will be polynomially, rather than exponentially, large in n:

Corollary 2. *If at most k of F's n vertices are visible from any point in F, then the graph G_I has $O(n^5 2^k)$ nodes.*

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 3. *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 cell r in P_ϕ then, by Lemma 6, 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 cell 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 cell 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 cell 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 6 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. $\square$

5 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 computes trajectories only for the case of $\phi = \pi$. In this case, the roadmap in P_ϕ 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.

Contrary to the description given above, we do not actually construct all of G_I prior to the search, but rather construct it on the fly, adding nodes and edges as necessary. This technique provides a significant speedup, for the full graph contains many irrelevant information states, as well as some unreachable ones.

We now present some computed examples.[6] In all the figures, light gray 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 Figures 5 & 12 are office-like environments composed of

[6]Animations of these and other computed examples can be found at: **http://ai.stanford.edu/$\sim$gerkey/research/pe**, and in Appendix A.

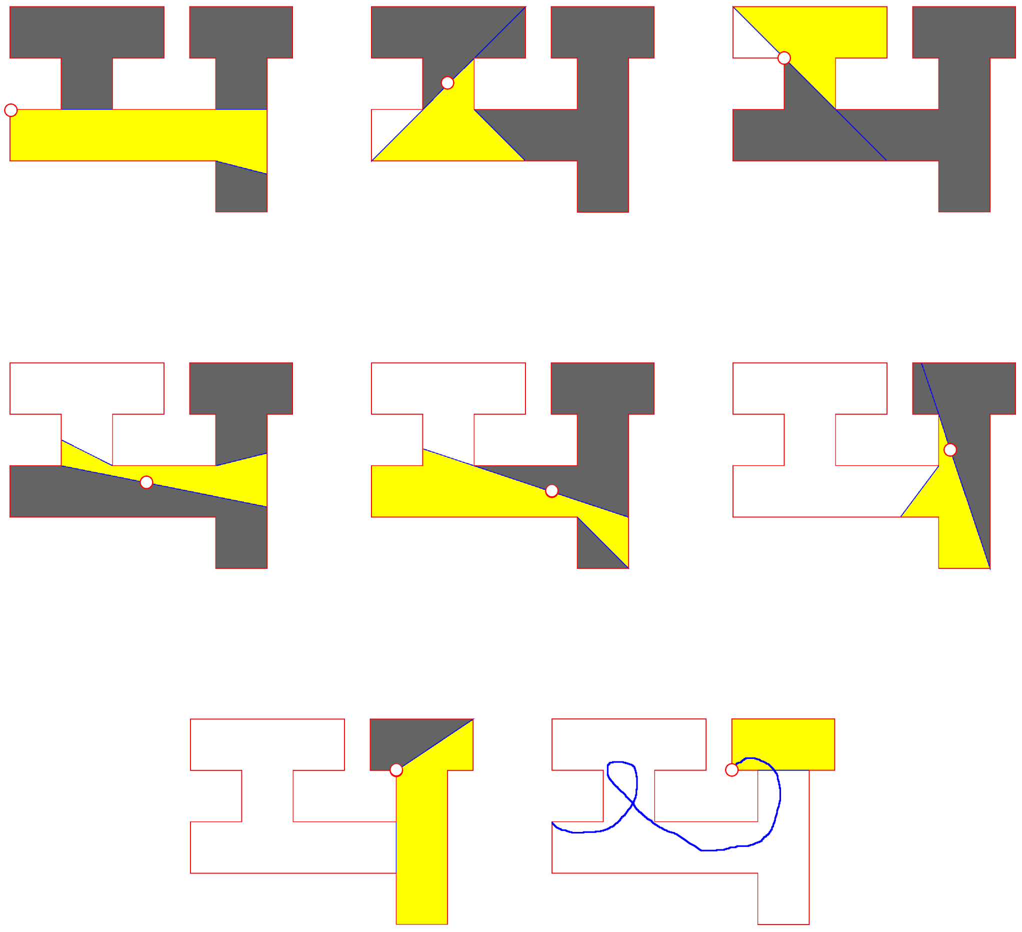

Figure 12: *A computed clearing trajectory for a π-searcher in an office-like environment (see also Extension 2).*

hallways connecting rooms. Shown in Figure 15 is a more complex environment, for which the computed trajectory had 42 steps.

6 Extensions and further examples

Since we may want to search environments that cannot be searched by a single ϕ-searcher, it is important to consider how we can extend this algorithm to handle multiple searchers. One way is to include in the construction of G_I the joint configuration space 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 cell boundary in P_ϕ. It is easy to construct environments in which two searchers will clear a portion of the environment without realizing it (see Guibas et al. (1999) for an example). On the other hand, we have not yet encountered an environment for which this

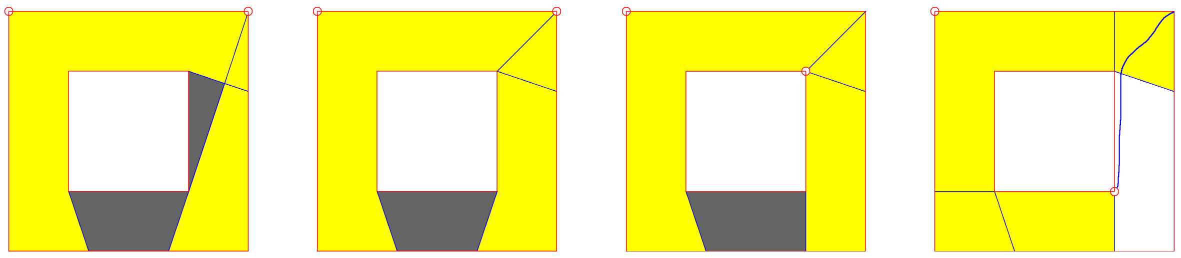

Figure 13: *This example, with one loop, is the simplest environment that requires two π-searchers to clear. In this case, the searcher in the upper left corner remains stationary, watching the upper and left corridors, while the other searcher moves to clear the right and lower corridors (see also Extension 3).*

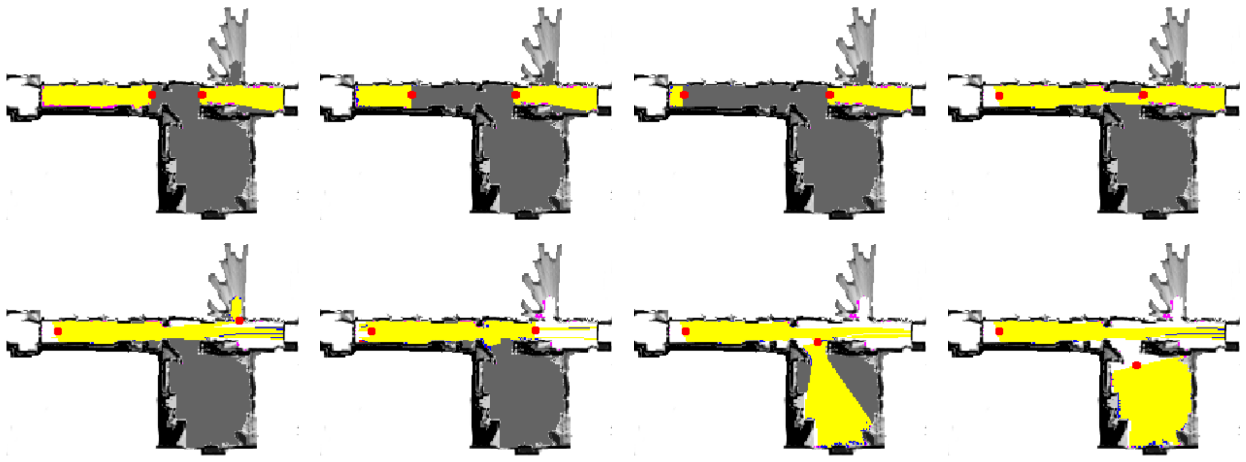

Figure 14: *A computed clearing trajectory for two π-searchers in a sensor-based map of a corridor with two adjoining rooms. The trajectory was computed from a manually-generated floorplan-style approximation of this sensor-based map. In this case, one searcher moves to the left end of the hall, and turns around to cover the hall while the other searcher sequentially clears the two rooms. This trajectory was executed by two Pioneer mobile robots (see also Extension 4).*

incompleteness actually causes the algorithm to fail to find a solution when one exists, and we believe that such cases are rare.

A more serious drawback to extending our algorithm to multiple searchers in this way is that the joint configuration space grows 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; shown in Figure 13 is a simple loop 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 to compute this solution, in which two searchers work together to clear the environment. It is possible to restore completeness by constructing a new information graph that accurately reflects the possible changes in information induced by multiple searchers, but this change would further exponentially increase the problem difficulty.

There are other avenues for extension that concern the capabilities of robots. For example, there is no physical sensor that can provide target detection over an unlimited range. If the searcher's sensor range is short enough to make a difference in a given environment (e.g., a laser range-finder in a building with very long corridors), then we must account for the range, or risk generating non-solution trajectories. We can incorporate into the algorithm a limited range sensor (or indeed, any sensor model) by modifying the step in which the contamination map is updated during the construction of the information graph. The limited-range ϕ-searcher will have a different visibility polygon, which will induce different map. Again, this extension comes at the expense of completeness, for there are now distance-specific visibility constraints that are not captured in our cell decomposition.

Another extension is to allow for non-holonomic constraints on the searcher. Consider for example, a two-wheel differentially-driven robot, which can translate forward and backward and rotate in place, but cannot translate sideways. One way to account for this constraint is to modify the construction of the information graph to allow only moves that can be executed by a differentially-driven robot. Then any computed trajectory will be a feasible solution. We could go further with this idea by restricting the robot to move only within the FOV of its evader sensor. If the the robot uses the same sensor for obstacle avoidance as for evader detection (this is often the case), such a restriction would make for more robust solutions by never requiring the robot to drive blindly, which would force it to rely on the accuracy of the map for collision avoidance.

Figure 14 demonstrates the application of this modified algorithm to a team of two laser-equipped Pioneer mobile robots (Figure 1(a)) searching a hallway with two adjoining rooms. In this case, we manually generated a floorplan-style approximation of the learned sensor-based map shown in the figure (the robots used the original map for localization during the trajectory execution). We then planned for two searchers in the resulting, simplified map, using the pruned configuration space described above. The resulting cooperative strategy was executed on the robots: one robot covered the hallway while the other searched the two rooms in succession.

Not surprisingly, these extensions also sacrifice completeness. A complete algorithm for a non-holonomic searcher would have to answer the following non-trivial motion planning question: given a robot with non-holonomic constraints and a fixed sensor orientation, does there exist a feasible trajectory from pose A to pose B such that a set S of points remain in view en route?

7 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 algorithms 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 configuration space, and gave examples of computed trajectories for single searchers and for teams of two searchers. While the algorithm applies to ϕ-searchers with arbitrary fields of view, for reasons of simplicity and efficiency we restricted our implementation to the special case of $\phi = \pi$.

Planning in the joint configuration 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 (Gerkey et al. 2005). 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 configuration 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

Adler, M., Räcke, H., Sivadasan, N., Sohler, C. & Vöcking, B. (2003), ‘Randomized Pursuit-Evasion in Graphs’, *Combinatorics, Probability and Computing* **12**(3), 225–244.

Avnaim, F., Boissonat, J. D. & Faverjon, B. (1988), A practical exact planning algorithm for polygonal objects amidst polygonal obstacles, *in* ‘Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)’, pp. 1656–1660.

Burgard, W., Fox, D., Moors, M., Simmons, R. & Thrun, S. (2000), Collaborative multi-robot exploration, *in* ‘Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)’, San Francisco, California, pp. 476–481.

Burnikel, C., Fleischer, R., Mehlhorn, K. & Schirra, S. (1999), Efficient exact geometric computation made easy, *in* ‘Proc. of the ACM Symp. on Computational Geometry’, Miami Beach, Florida, pp. 341–350.

Gerkey, B. P., Thrun, S. & Gordon, G. (2005), Parallel stochastic hill-climbing with small teams, *in* L.E.Parker et al., eds, ‘Multi-Robot Systems: From Swarms to Intelligent Automata, Volume III’, Springer, the Netherlands, pp. 65–77.

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

Guibas, L. J., Motwani, R. & Raghavan, P. (1995), The robot localization problem, *in* K. Goldberg, J.-C. Latombe, R. Wilson & D. Halperin, eds, ‘Algorithmic Foundations of Robotics’, A.K. Peters, Natick, Massachusetts, pp. 269–282.

Hájek, O. (1975), *Pursuit Games*, Academic Press, New York.

Isaacs, R. (1965), *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*, John Wiley & Sons, New York.

Isler, V., Kannan, S. & Khanna, S. (2005), ‘Randomized pursuit-evasion in a polygonal environment’, *IEEE Transactions on Robotics* **21**(5), 875–884.

Jung, B. & Sukhatme, G. S. (2002), ‘Tracking Targets using Multiple Robots: The Effect of Environment Occlusion’, *Autonomous Robots* **13**(3), 191–205.

Kalra, N., Ferguson, D. & Stentz, A. (2005), Hoplites: A market-based framework for planned tight coordination in multirobot teams, *in* ‘Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)’, Barcelona, Spain, pp. 1170–1177.

Koenig, S. & Simmons, R. G. (1993), Exploration with and without a map, *in* ‘Proceedings of the AAAI Workshop on Learning Action Models at the Eleventh National Conference on Artificial Intelligence (AAAI)’, pp. 28–32. (also available as AAAI Technical Report WS-93-06).

Latombe, J.-C. (1991), *Robot Motion Planning*, Kluwer Academic Publishers, Norwell, Massachusetts.

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

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

Leven, D. & Sharir, M. (1987), 'An efficient and simple motion planning algorithm for a ladder moving in a 2-dimensional space amidst polygonal barriers', *J. of Algorithms* **8**, 192–215.

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

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

O'Rourke, J. (1987), *Art Gallery Theorems and Algorithms*, Oxford University Press, New York.

Park, S.-M., Lee, J.-H. & Chwa, K.-Y. (2001), Visibility-Based Pursuit-Evasion in a Polygonal Region by a Searcher, *in* F. Orejas and P.G. Spirakis and J. van Leeuwen, ed., 'Automata, languages and programming', Lecture Notes in Computer Science 2076, Springer-Verlag, Berlin, pp. 456–468.

Parker, L. E. (1999), 'Cooperative Robotics for Multi-Target Observation', *Intelligent Automation and Soft Computing* **5**(1), 5–19.

Parsons, T. (1976), Pursuit-evasion in a graph, *in* Y. Alavi & D. Lick, eds, 'Theory and Applications of Graphs', Lecture Notes in Mathematics 642, Springer-Verlag, Berlin, pp. 426–441.

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

Schwartz, J. & Sharir, M. (1983), 'On the 'piano movers' problem: I. the case of a rigid polygonal body moving amidst polygonal barriers', *Communications on Pure and Applied Mathematics* **36**, 345–398.

Shermer, T. C. (1992), 'Recent results in art galleries', *Proceedings of the IEEE* **80**(9), 1384–1399.

Spletzer, J. R. & 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), PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Suzuki, I. & 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. & 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. & Matarić, M. J. (2001), Broadcast of Local Eligibility for Multi-Target Observation, *in* L. E. Parker, G. Bekey & J. Barhen, eds, 'Distributed Autonomous Robotic Systems 4', Springer-Verlag, New York, pp. 347–356.

Yamauchi, B. (1998), Frontier-Based Exploration Using Multiple Robots, *in* 'Proc. of Autonomous Agents', Minneapolis, Minnesota, pp. 47–53.

A Index to Multimedia Extensions

The multimedia extensions to this article can be found online by following the hyperlinks from www.ijrr.org:

Extension	**Media type**	**Description**
1	Video	Example 1: One π-searcher clears two rooms connected by a hallway.
2	Video	Example 2: One π-searcher clears a U-shaped office-like environment.
3	Video	Example 3: Two π-searchers clear a loop.
4	Video	Example 4: Two laser-equipped Pioneer robots clear two rooms connected by a hallway.
5	Video	Example 5: One π-searcher clears an environment with four alcoves.
6	Video	Example 6: One π-searcher clears three rooms connected by a hallway.

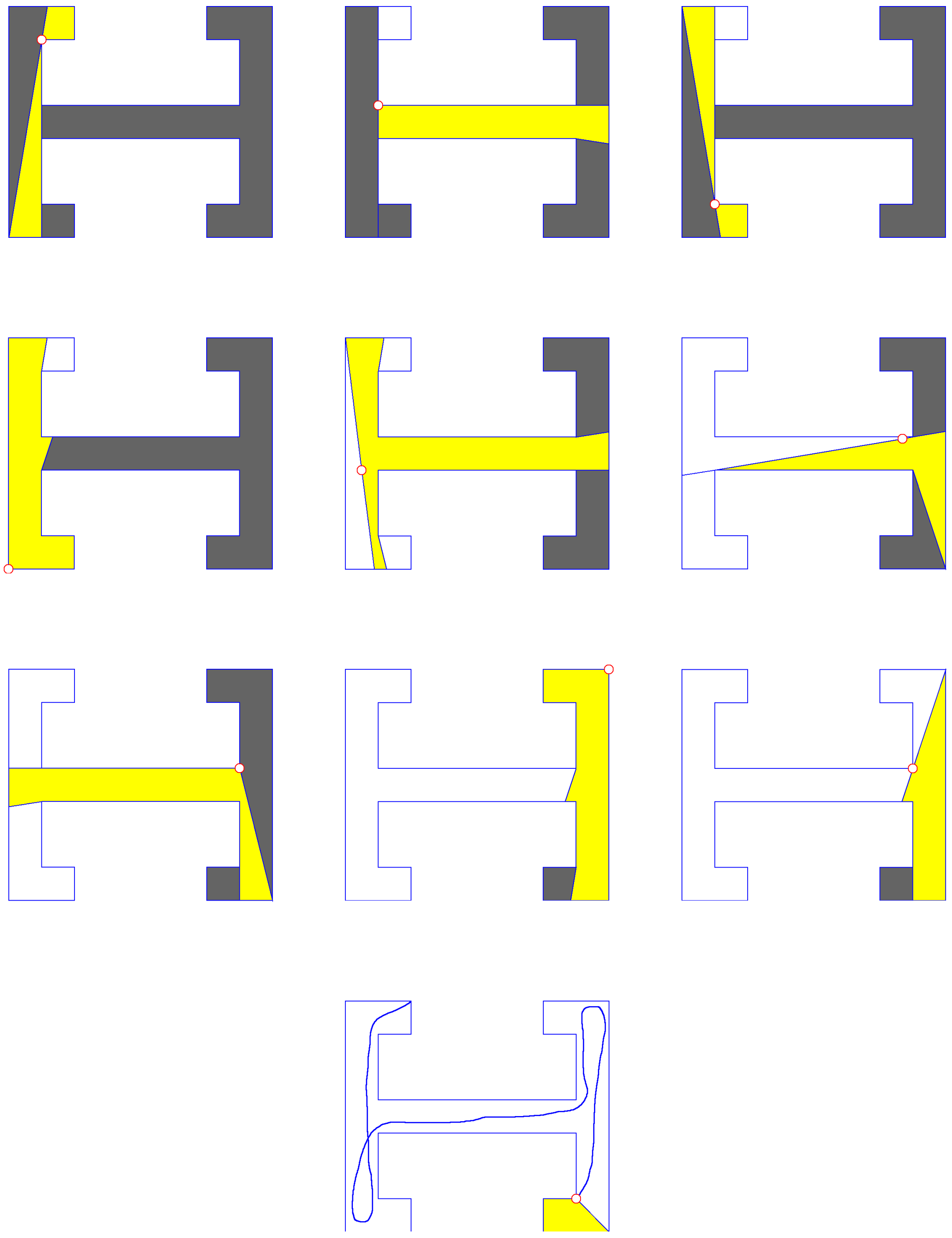

Figure 15: *In this more complicated example, the π-searcher first clears the left hand side from top to bottom, then moves through the horizontal hallway to clear the right hand side, from top to bottom (see also Extension 5).*