Persistent pursuit-evasion: the case of the preoccupied pursuer

Nicholas M. Stiffler[1] Andreas Kolling[2] Jason M. O'Kane[1]

***Abstract*— We consider a visibility-based pursuit-evasion problem in which a single robot with an omnidirectional but unreliable sensor moving through an environment must systematically search that environment to detect an unpredictably moving target. A common assumption in visibility-based pursuit-evasion is that the sensors used to detect the evader are perfectly reliable. That is, any evader that moves within view of the pursuer for any interval of time will be detected. This assumption is problematic because, when implemented on real sensor systems, such plans cannot account for the possibility of short-term false negative errors in evader detection. This paper addresses this limitation by introducing a model based on the idea of *pessimal unoccluded distance* to reason about the degree of plausibility that the evader may be concealed within each occluded region. We describe a decomposition of the environment that fully characterizes the opportune moment for an evader to take advantage of sensor error. Furthermore, we present a complete algorithm that solves the active problem of planning a search for a pursuer which maximizes the distance that the evader must travel through the pursuer robot's sensor footprint.**

I. Introduction

Pursuit-evasion games are a family of problems involving two groups of agents, pursuers and evaders. The pursuers' goal is to locate all of the evaders by systematically searching the environment. The evaders' goal is to remain undetected by the pursuers. Though these games are reminiscent of the children's games of tag, cops-and-robbers, and hide-and-seek, they also have practical applications for search-and-rescue [6], [28], surveillance [1], [19], and missile-guidance systems [9], [23].

Current solutions to many pursuit-evasion problems have several drawbacks that are frequently encountered in practical scenarios. In particular, many of the plans generated by existing solutions are not robust to sensor error. These solutions assume that the sensors used to detect evaders are perfectly reliable. This assumption is problematic because, when implemented on real sensor systems, such plans cannot account for the possibility of short-term false negative errors in evader detection.

One of the major advantages of existing solutions to pursuit-evasion problems for errorless sensors is their ability to generate a winning strategy for the pursuer or pursuers, or to determine that no such strategy exists [7], [27]. A naïve approach to overcoming false negative errors in this context is simply to "Try again"—to repeatedly execute the plan generated by one of the preexisting solutions. Assuming that false negatives are distributed independently, such a plan would eventually succeed in locating the evaders. The downfall of this approach centers around the underutilization of information gained during previous executions. Presently, these solutions lack the ability to incorporate the information gained from one execution cycle to form more effective plans for successive iterations.

This paper considers a geometric formulation of the pursuit-evasion problem that incorporates potential sensor error. The idea is to model the possibility of sensor failures by measuring the worst-case distance that an evader would have to travel under the pursuer's sensor footprint to remain undetected. This approach sidesteps the probabilistic modeling burden found in graph-based formulations of the pursuit-evasion problem [12] as well as the more general study of *optimal search* [30], and is applicable to any environment representation in which distance between non-visible regions can be computed.

Rather than having a fixed termination time, the resulting paths define a pursuer position for *any* time $t \in [0, \infty)$. We call such paths *persistent pursuit-evasion paths* because they guarantee that the pursuer will be within a line-of-sight of the evader infinitely often.

After a brief tour of related research in Section II, this paper makes several novel contributions.

1) Section III formally defines the persistent pursuit-evasion problem, introducing the notion of the *pessimal unoccluded distance* of a pursuer path.
2) Section IV describes a passive algorithm for computing the pessimal unoccluded distance of a given pursuer path. The algorithm is based on a new kind of planar decomposition we call the *jump decomposition*, which enables updates to be performed at only finitely many points along the pursuer's path.
3) Section V shows how to use this passive algorithm to *generate* pursuer paths that drive the pessimal unoccluded distance arbitrarily high. This active algorithm is based on a forward search through the jump decomposition.
4) Section VI describes an evaluation of both the passive and active algorithms in simulation.

We close by discussing future research (Section VII).

This material is based upon work supported by the National Science Foundation under Grant No. 0953503. [1]N. M. Stiffler and J. M. O'Kane are with the Department of Computer Science and Engineering, University of South Carolina, 301 Main St., Columbia, SC 29208, USA. `{stifflen, jokane}@cse.sc.edu`
[2]A. Kolling is with the iRobot Corporation, Pasadena CA, USA `andreas.kolling@gmail.com`

II. Related Work

This work can be viewed as an intertwining between pursuit-evasion problems, optimal search, and persistent monitoring and surveillance problems.

A. Pursuit-Evasion

The first visibility-based pursuit-evasion problem [31] was proposed as an extension of the watchman route problem [2] and is a geometric formulation of the traditional graph-based pursuit-evasion problem [21], [22]. Research on the visibility-based pursuit-evasion problem has produced numerous results for both the single pursuer and multiple pursuer variants of the problem.

For the single pursuer visibility-based pursuit-evasion problem, a complete solution [7], a randomized solution [8], and an optimal shortest path solution [28] have been found.

The capture condition for the general visibility-based pursuit-evasion problem is defined as having an evader lie within the pursuer's capture region. There has been substantial research focused on how the visibility-based pursuit-evasion problem changes when a robot has different capture regions. The k-searcher is a pursuer with k visibility beams [31], the ∞-searcher is a pursuer with omni-directional field of view [7], and the ϕ-searcher is a pursuer whose field of view [5] is limited to an angle $\phi \in (0, 2\pi]$. Note that all of these approaches consider evaders with unbounded speed.

Others have studied scenarios in which there are additional constraints, such as the case of a curved environment [15], an unknown environment [25], a maximum bounded speed for the evader [33], or constraints on the pursuer similar to those of a typical bug algorithm [24].

As a result of the problem complexity [29], there is a wide range of literature with differing techniques attempting to solve the multi-robot visibility-based pursuit-evasion problem. Some recent results involve using some of the pursuers as stationary sentinels while other pursuers continue with the search [13]. Another approach involves maintaining complete coverage of the frontier [4]. There are other variants of the pursuit-evasion problem where the pursuers are teams of unmanned aerial vehicles [11].

B. Optimal Search

Optimal search [30] utilizes a Bayesian approach for maintaining a target distribution and uses that information for guiding the planning of optimal search paths. The essential idea from optimal search that translates to our pursuit-evasion domain is that a search would emphasize those areas of the environment with the highest probability of containing a target. Optimal search algorithms then allow for the prediction of the no-detection likelihood which is the probability that the target will remain undetected at a given instant of the search.

Yu and LaValle [35] present a mechanism for maintaining a distribution of hidden targets during a search in the

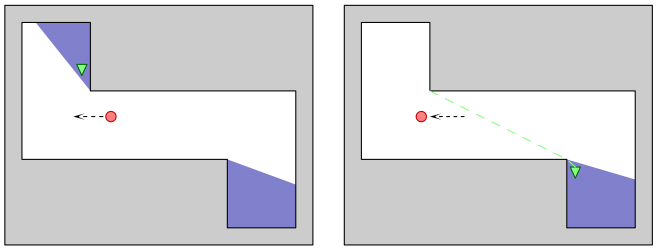
Fig. 1. An illustration of our motion model for the evader that occurs when a pursuer (red circle) causes the evader (green triangle) to "jump" (travel under the pursuer's sensing region) from one shadow to the another.

presence of probabilistic uncertainty that can be generalized from the more general target tracking case to into the pursuit-evasion domain. Extending this framework which provides passive update rules to the active problem of generating a pursuer search path is made increasingly difficult because of the complexity involved in generating a general probability model that accurately reflects the evader behavior. The problem becomes tenable once some assumptions as to the behavior of the evaders are known [27].

Our approach can avoid filtering entirely by reasoning about the most pessimistic eventuality where an evader can decide to exploit an instantaneous sensing error to prolong its capture. The intuition is that it will become increasingly unlikely that an evader will continue to take full advantage of any sensing failure.

C. Persistent Monitoring and Surveillance

Persistent monitoring and surveillance tasks require a tracker or team of trackers to perform their monitoring/surveillance task in perpetuity. This idea has been extended to graphs where policies are planned for an agent or team of agents tasked with patrolling nodes to intercept an attacker [10], [16]. The "perpetuity" aspect makes these problems well-suited to be carried out by a robot or robot team [18]. Visibility-based monitoring problems commonly occur in many applications such as security and surveillance [32], infrastructure inspection [20], and environmental monitoring [26]. The increased availability of mobile robots capable of performing these tasks has led to heightened interest [14], [17], [34] in recent years.

III. Problem Statement

This section formalizes the visibility-based pursuit-evasion problem considered in this paper. We begin by describing the model used to represent the environment, evader, and pursuer (Section III-A) and then discuss the criterion that must be satisfied to classify the pursuer's motion as a persistent pursuit strategy (Section III-B).

A. Representing the Environment, Pursuer, and Evader

The environment is a polygonal free-space, defined as a simply-connected closed and bounded set $W \subset \mathbb{R}^2$, with a polygonal boundary ∂W composed of n vertices.

A pursuer moves to locate the evader. We assume that the pursuer knows W. Therefore, from a given start position, the pursuer's motions can be described by a continuous function $p : [0, \infty) \to W$, so that $p(t) \in W$ denotes the position the pursuer at time $t \geq 0$. The function p is called a *motion strategy* for the pursuer.

The pursuer carries a sensor that can detect the evader. The sensor is omnidirectional and has unlimited range, but cannot see through obstacles. For any point $q \in W$, let $V(q)$ denote the visibility region at point q, which consists of the set of all points in W that are visible from point q. That is, $V(q)$ contains every point that can be connected to q by a line segment in W. Note that $V(q)$ is a closed set.

For any $q \in W$, consider the boundary of $V(q)$. The edges of this boundary are either along ∂W or belong to an occlusion ray. An **occlusion ray** is a ray starting at the point r extending in direction $(r - q)$, denoted $\mathrm{ray}(r, r - q)$.

Informally, an occlusion ray originating at point q is a ray that acts as a boundary separating a visible and non-visible portions of W.

The evader is modeled as a point that can translate within the environment. Let $e(t) \in W$ denote the position of the evader at time $t \geq 0$. The path e is a continuous function $e : [0, \infty) \to W$, in which the evader is capable of moving arbitrarily fast (i.e. a finite, unbounded speed) within W. The evader trajectory e is unknown to the pursuer. Without loss of generality we assume that there is a single omniscient evader.

We assume that the time spent by the evader in the pursuer's visibility region is negligible. That is, when the evader enters the pursuer's visibility region, we assume that the evader moves immediately to some other location outside of $V(p(t))$. We refer to this behavior as a *jump*, with the intuition that if the evader is detected, it will immediately 'jump' beyond the pursuer's visibility region. Figure 1 illustrates this behavior. Since the evader has unbounded velocity, we treat the time for this motion as negligible and consider only the *distance* traveled by the evader within the pursuer's visibility region.

Definition 1: For given pursuer and evader trajectories p and e and a given finite time t, the total distance travelled by the evader through the pursuer's visibility region up to time t is referred to as the **unoccluded distance**, denoted $\mathrm{ud}(p, e, t)$.

The intuition is that the unoccluded distance measures the amount of travel within $V(p(t))$ that the evader makes. As this value increases, the pursuer has an increased opportunity to detect the evader.

Definition 2: For a given pursuer trajectory p and a finite time t we define the **pessimal unoccluded distance** for that pursuer trajectory at that time as the worst-case unoccluded distance over all evader trajectories:

$$\mathrm{pud}(p, t) = \min_{e} \mathrm{ud}(p, e, t). \tag{1}$$

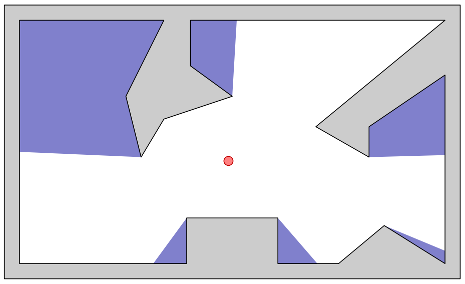

Fig. 2. An environment with a pursuer (red circle) and six shadows (filled path-connected regions).

B. Persistent Search Strategies

The pursuer's goal is to drive $\mathrm{pud}(p, t)$ arbitrarily high as t increases.

Definition 3: A pursuer trajectory p is called **persistent** if, for any $d \in [0, \infty)$, there exists some t for which $\mathrm{pud}(p, t) \geq d$.

The intuition is that a persistent strategy is one in which the evader is forced to cross through the pursuer's sensor footprint repeatedly as time passes.

IV. Passive PUD Updates

This section considers the passive problem of computing the pessimal unoccluded distance given a pursuer path p, an evader path e, and a finite time horizon t. We first give a formal definition for the areas of the environment hidden from the pursuer, called shadows (Section IV-A). We then describe a specialized decomposition of the environment that identifies, along the given pursuer path, a finite locus of points at which this update rule must be applied (Section IV-B). We then present a method for updating the unoccluded distance based on potential evader jumps (Section IV-C).

A. Shadows and Spuds

The key difficulty in locating the evader is that the pursuer cannot, in general, see the entire environment at once. This section contains some definitions for describing and reasoning about the portion of the environment that is not visible to the pursuer at any particular time.

Definition 4: The portion of the environment not visible to the pursuer at time t is called the **shadow region** $\mathcal{S}(t)$, defined as

$$\mathcal{S}(t) = W - V(p(t))$$

Note that the shadow region may contain zero or more nonempty path-connected components, as seen in Figure 2.

Definition 5: A **shadow** is a maximal path-connected component of the shadow region.

A shadow is often referenced by the environment reflex vertex that induces the occlusion ray that serves as the shared boundary between the pursuer's visibility region and the corresponding shadow. This reflex vertex is henceforth referred to as the *anchor* of the shadow. The intuition is that a shadow can potentially change as the occlusion ray rotates around its anchor, but will remain anchored that a particular vertex.

Notice that $\mathcal{S}(t)$ is the union of the shadows at time t. The important idea is that, except for the negligible time intervals during which it is visible to the pursuer, the evader is always contained in exactly one shadow, within which it can move freely.

For our pursuit-evasion problem, the crucial piece of information about each shadow at each time is the worst-case unoccluded distance the evader could achieve for a path currently within that shadow. We can assign a nonnegative real number label to each shadow which corresponds to this distance.

Definition 6: For an individual shadow s_i, a pursuer path p, and a time t, the **shadow pessimal unoccluded distance**, abbreviated **spud**, is defined as

$$\operatorname{spud}(p, s_i, t) = \operatorname*{argmin}_{e \in \mathcal{P}(s_i)} \operatorname{ud}(p, e, t), \tag{2}$$

in which $\mathcal{P}(s_i)$ denotes the set of all evader paths $e : [0, t] \to W$ for which $e(t) \in s_i$.

Using this worst-case reasoning, we can completely represent the pursuer's progress in searching for the evader by the pursuer's current position and the current collection of spuds.

Lemma 1: For any pursuer trajectory p and any time t, we have

$$\operatorname{pud}(p, t) = \min_{s_i} \operatorname{spud}(p, s_i, t),$$

in which the minimum is over all shadows at time t.

Proof: Follows directly from Equations 1 and 2. ■

The pursuer's goal is to increase each of the spuds in order to increase the overall pessimal unoccluded distance. The intuition is that an evader that behaves according to the motion model described in Section III-A will be hiding in exactly one of the shadows, though the identity of the exact shadow concealing the evader is unknown to the pursuer. By reasoning about the minimum spud, the pursuer can compute the overall pessimal unoccluded distance value achieved by its movements.

B. Jump decomposition: Sufficient sets of jump locations

This section analyzes the locations at which the evader may jump from one shadow to another. Specifically, we describe a decomposition of W into a finite collection of polygonal cells, called the jump decomposition, and show that it is sufficient to consider evader jumps only when the pursuer crosses a cell boundary in the jump decomposition, or when the pursuer changes direction. The jump decomposition is a refinement of the visibility cell decomposition used by Guibas, Latombe, LaValle, Lin, and Motwani [7], inserting additional cell divisions to account for local minima in the distance that the evader would need to jump.

Definition 7: The jump decomposition (Figure 3) is formed by extending three types of rays.

1) Obstacle edges are extended in either direction, or both directions if possible. These rays originate at the reflex vertices of the environment and extend into the interior of W (Figure 3, green rays).

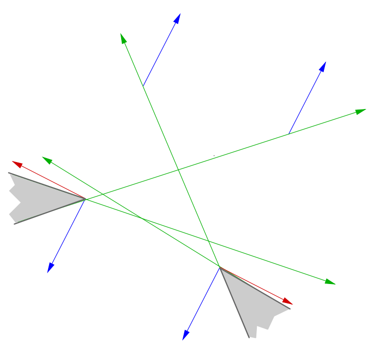
Fig. 3. Jump decomposition between two environment vertices illustrating the three types of rays used to partition the environment.

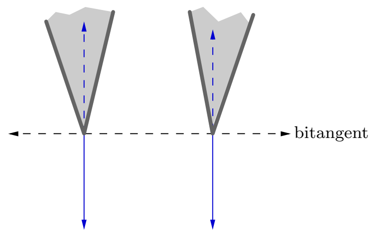

Fig. 4. A scenario where only one pair of rays corresponding to Item 3 of our jump decomposition exists. The other pair consists of rays who are immediately in collision with the environment, thus they can be safely discarded.

2) Pairs of mutually visible environment vertices are extended outwards only if both directions are free along the bitangent line through the pair of points (Figure 3, red rays).
3) Rays are extended outwards from pairs of mutually visible environment reflex vertices, orthogonal to the line segment connecting the vertices. On either side of the line connecting the vertices, these rays are extended only if both rays on that side have nonzero length (Figure 4). The start points of these rays correspond to the projection of the vertices onto the inflection ray (Item 1) of the partnering vertex (Figure 3, blue rays).

This decomposition is valuable because its cell divisions show the opportune locations for the evader to jump from one shadow to another. The next lemma makes this idea more precise.

Lemma 2: For any positive real number a, any piecewise linear pursuer path p, and any time t, if there exists an evader path e for which $\operatorname{ud}(p, e, t) = a$, then there also exists an evader path e' with $\operatorname{ud}(p, e', t) \leq a$, in which e' jumps between shadows only when the pursuer crosses between regions of the decomposition, or when the pursuer changes direction.

Proof: Construct e' as identical to e, except that any jumps made by e are delayed, if necessary, until a combinatorial change occurs in the shadow set, or until the shortest path between some set of shadows reaches a local minimum along the pursuer's path. Clearly $\operatorname{ud}(p, e', t) \leq \operatorname{ud}(p, e, t)$, since the same jumps are made by e' and e, possibly with smaller unoccluded distances for each. It remains to show that e' jumps only when the pursuer crosses between cells in the jump decomposition or when

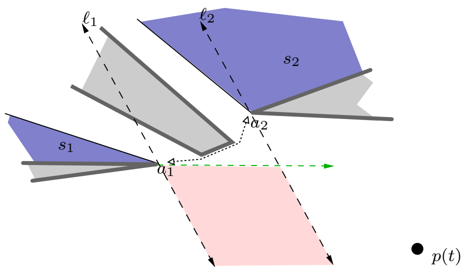

Fig. 5. Scenario where an obstacle inhibits the evader from directly jumping along the projection of an anchor to an occlusion ray. Note that if an obstacle crosses the bitangent between a_1 and a_2 then the jump point reverts to the anchors.

the pursuer changes direction.

It is well known that combinatorial changes to the set of shadows occur only when the pursuer is on an edge in the visibility cell decomposition. The jump decomposition includes these edges as Items 1 and 2 in Definition 7.

Finally, we identify the locations at which local minima of the distance between a pair of shadows s_1 and s_2, with anchor vertices a_1 and a_2 respectively, can occur. The intuition is that the shortest path between any pair of shadows occurs either between the shadows' anchors, or between one anchor and a non-anchor point along the opposite occlusion ray.

Let ℓ_1 denote the line passing through a_1, perpendicular to line passing through a_1 and a_2. Similarly, ℓ_2 passes through a_2 and is parallel to ℓ_1. For the distance between these two shadows, there are three cases, depending on the location of $p(t)$ relative to ℓ_1 and ℓ_2.

1) If $p(t)$ is between ℓ_1 and ℓ_2, or if a_1 is not visible from a_2 (Figure 5), then the jump distance between s_1 and s_2 is the shortest path in W from a_1 to a_2. Note that this only applies for pursuer positions at which s_1 and s_2 exist, which occurs when $p(t)$ is on the appropriate side of the Item 1 ray extensions from a_1 and a_2. See Figure 6.
2) If $p(t)$ is *not* between ℓ_1 and ℓ_2, and $p(t)$ is closer to ℓ_2 than ℓ_1, then the jump distance between s_1 and s_2 is the length of the shortest path in W from a_1 to $\mathrm{ray}(a_2, p(t) - a_2)$. Moreover, for any linear path segment within this region, this distance varies monotonically. See Figure 7.
3) Finally, if $p(t)$ is not between ℓ_1 and ℓ_2, and $p(t)$ is closer to ℓ_1 than ℓ_2, the same argument applies *mutatis mutandis*.

Note that, within each of these cases, the jump distance between two shadows cannot reach a local minimum, unless the pursuer changes direction. Moreover, the jump decomposition includes edges separating these three cases from one another.

Therefore, e' jumps only when the pursuer changes direction or crosses to a new cell in the jump decomposition. ■

C. Spud updates

The intuition of Lemma 2 is that, to maintain the pessimal unoccluded distance $\mathrm{pud}(p, t)$ as the pursuer

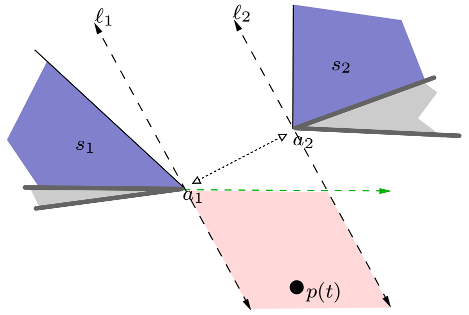

Fig. 6. Illustration of the instance where the shortest distance between two shadows occurs at the anchors. Notice, that the ray on ℓ_2 does not begin at a_2. This occurs because of the inflection (dashed green ray) coming from a_1, if the pursuer moves beyond the inflection then shadow s_1 will *disappear*.

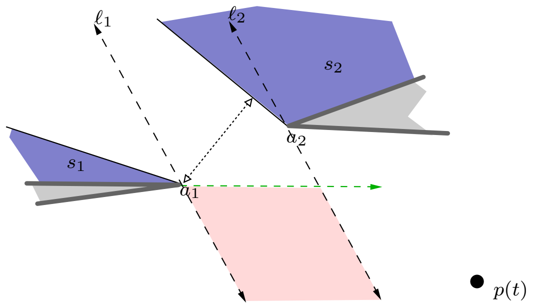

Fig. 7. Illustration of the scenario where the shortest distance between two shadows passes through an anchor and an occlusion ray. Notice, that since $p(t)$ is not between ℓ_1 and ℓ_2, and $p(t)$ is closer to ℓ_2, the jump occurs from the occlusion ray at a_2 to the anchor vertex a_1.

moves, we can update the spuds a finite number of times, only when the pursuer changes directions or crosses a cell boundary of the jump decomposition.

To accomplish this, we compute the jump decomposition as a doubly-connected edge list [3], and associate with each interior half-edge h, a possibly empty set of jumps

$$J(h) = \{s_{\mathrm{src}_1} \to s_{\mathrm{tgt}_1}, \dots, s_{\mathrm{src}_m} \to s_{\mathrm{tgt}_m}\}$$

that may occur when the pursuer crosses that half-edge. For example, for a half-edge h corresponding to the disappearance of a shadow s, we include elements in $J(h)$ representing jumps from s to each other shadow in $\mathcal{S}(t)$. If h instead corresponds to the appearance of a new shadow in a region that was previously fully visible, we initialize the spud of that shadow to ∞ (indicating that, without a jump, the evader cannot be in that region), and include elements in $J(h)$ representing jumps *to* the new shadow from each other shadow. Likewise, we associate with each interior face f a possibly empty set of jumps

$$J(f) = \{s_{\mathrm{src}_1} \to s_{\mathrm{tgt}_1}, \dots, s_{\mathrm{src}_m} \to s_{\mathrm{tgt}_m}\}$$

that may occur when the pursuer changes direction within f. These are jumps between precisely the pairs of shadows for which Cases 2 or 3 in the proof of Lemma 2 apply.

In either case—crossing to a new cell in the jump decomposition, or changing direction within the interior one of its faces—we update the spuds by seeding a queue Q with the associated jumps and executing Algorithm 1. This algorithm accounts for the possibility of multiple

Algorithm 1 UPDATE_SPUDS$(p(t), J)$

```
Input: A pursuer position $p(t)$
Input: A set of plausible jumps $J$
Input: Initial spuds $\text{spud}(p, s_i, t)$ for each shadow
Output: Updated spud values.

  $Q \leftarrow$ queue initialized with the jumps in $J$
  while $Q$ is not empty do
    $(s_{\text{src}} \to s_{\text{tgt}}) \leftarrow Q.\text{pop}()$
    if $\text{spud}(p, s_{\text{src}}) + \text{dist}(s_{\text{src}}, s_{\text{tgt}}) < \text{spud}(p, s_{\text{tgt}})$ then
      $\text{spud}(p, s_{\text{tgt}}) \leftarrow \text{spud}(p, s_{\text{src}}) + \text{dist}(s_{\text{src}}, s_{\text{tgt}})$
      for each shadow $s$ at time $t$ do
        $Q.\text{insert}(s_{\text{tgt}} \to s)$
      end for
    end if
  end while
```

jumps in rapid succession using a Dijkstra-like propagation of the spuds.

V. Generating Persistent Pursuit Strategies

In this section, we describe our algorithm for actively computing a persistent pursuer strategy for a given environment. Since the strategy should continue indefinitely, we describe the algorithm as coroutine that periodically emits segments of the strategy to be executed by the robot's lower-level control system. The underlying structure of our algorithm is a forward search on the region graph induced by the jump decomposition. It computes a sequence of successive regions that correspond to areas of the environment that the pursuer can visit to drive the pessimal unoccluded distance arbitrarily high.

A. Forward search

The algorithm works using a variation on the standard forward search. It maintains a priority queue Q of *search nodes*, each containing the following information:

- A finite-length path $p : [0, T] \to W$ representing a partial plan for the pursuer, up through time T, ending at the centroid of one of the interior faces of the jump decomposition.
- A reference to the face in the jump decomposition containing $p(T)$.
- The value $\text{spud}(p, s_i, T)$ for each shadow s_i, achieved by p, computing using the passive algorithm from Section IV.

The algorithm maintains a *target pessimal unoccluded distance* τ, and searches for a path whose pessimal unoccluded distance is strictly greater than τ. Initially, τ is set to 0.

The priority queue Q is ordered based on the length T of pursuer's path. We initialize Q with a single search node, in which $T = 0$, $p(0)$ is at the pursuer's initial state, and $\text{spud}(p, s_i, 0) = 0$ for every shadow. These initial spuds correspond to the assumption that the pursuer does not have any *a priori* information on the location of the evader.

At each iteration, the algorithm selects from Q the node (p, T, spud) with the smallest T value. It expands this node by appending to p path segments that travel to each neighboring interior face in the jump decomposition, updating the spuds using Algorithm 1. If those new nodes pass the validity test described below, they are inserted into Q.

The search continues in this manner until one of two conditions is satisfied.

1) If Q becomes empty, the algorithm has determined that no persistent pursuer strategy exists, and terminates accordingly.
2) If a node (p, T, spud) has $\text{pud}(p, T) > \tau$, we have found a path segment that increases the pessimal unoccluded distance above the target τ. When this occurs, we emit that path p to be executed, increase τ to $\text{pud}(p, T)$, and clear Q. The algorithm then resumes normally by generating the successors of the node (p, T, spud). In particular, the spuds achieved by p persist as the algorithm computes the next path segment to emit.

This process continues indefinitely, generating the infinite path p, one segment at a time.

B. Pruning

Prior to adding a new search node (p, T, spud) to the forward search queue Q, we perform a pruning operation to ensure that our search is making progress towards a solution by avoiding search nodes for which better alternatives have already been found.

Definition 8: A search node (p, T, spud) is **dominated** by another search node (p', T', spud') if

1) both $p(T)$ and $p'(T')$ reside in the same face of the jump decomposition, and
2) for every shadow s_i—Note that both $p(T)$ and $p'(T')$ induce the same set of shadows—we have
$$\text{spud}'(p', s_i, T') \;\leq\; \text{spud}(p, s_i, T).$$

The intuition is that if a node is dominated by another node, any future expansions that originate from the dominated node will be inferior to those generated by the dominating node.

The algorithm maintains a list of non-dominated search nodes for each interior face of the jump decomposition. We call a search node *valid* if it is not dominated by any node in the corresponding list. Each time we generate a new search node, we compare it with the existing list, and discard the new node if any existing nodes dominate it. If the node is valid, insert it into the list, and scan the list to delete any nodes dominated by the new one.

VI. Experiments

We have implemented the algorithm presented in Section V. This section presents several experiments to evaluate the performance of the algorithm.

As a basis to for comparison, we show not only our solution (denoted as SKO in Figures 8, 9, and 10) but also make a slight alteration to the algorithm presented by Guibas, Latombe, LaValle, Lin, and Motwani (hereafter, GL^{3}M) that is known to be complete for the case with no sensor error. The original GL^{3}M algorithm generates a path, starting from an initial position and ending elsewhere that guarantees that all of the evaders are captured. To apply this solution in the present context, we augment this solution by appending an additional path segment that returns to the initial position, and repeating this circuit indefinitely. The resulting path is a persistent pursuer strategy.

We performed simulations on three environments. For each one, we plot the pessimal unoccluded distance as a function of the distance traveled by the pursuer, measured in the abstract distance units employed by our custom simulator. In these plots, larger values indicate superior performance, corresponding paths that force the evader to jump longer distances.

Figure 8 shows a simple proof-of-concept environment. This case was chosen to illustrate the importance of the initial position to the GL^{3}M algorithm. The pursuer's start location was near the top-left corner of the environment. The GL^{3}M circuit requires additional travel to return to that area of the environment, whereas the path generated by our algorithm oscillates between left-of-center and right-of-center.

The environment in Figure 9 is an example from the literature [7]. This environment is known to require the pursuer to clear the corners of the environment using one of two different sequences:

1) Clear the right portion of the environment, followed by the top left portion, before finally clearing the bottom left portion.
2) Similarly, the pursuer can perform those clearing motions in reverse: bottom left, top left, right, to clear the environment.

We selected the center hallway connecting the left and right portions of the environment as the start point. Once again our algorithm outperformed the GL^{3}M path.

The environment in Figure 10 illustrates the effect that poor short term planning has on the global unoccluded distance. Due to the many branching hallways found in this environment, the pursuer must be careful determining the order in which these branches are explored.

VII. Conclusion

This paper described a novel approach to the visibility-based pursuit-evasion problem when dealing with sensing uncertainty due to potential false negative errors in the sensors used to detect the evader. Rather than modeling these errors probabilistically, we instead reasoned about the *pessimal unoccluded distance*, which is the worst-case distance that an evader would have to travel under the pursuer's sensor footprint. Our algorithm is designed to generate persistent pursuer paths that drive this value arbitrarily high. To that end, this paper made several novel contributions. We presented the jump decomposition which can be used in conjunction with a pursuer path to identify a finite number of instances where the evader's optimal motion might cross into the pursuer's sensing area. We then described an algorithm for the active problem of computing a persistent pursuer strategy. Our algorithm periodically emits partial plans that when appended together form a persistent pursuer strategy.

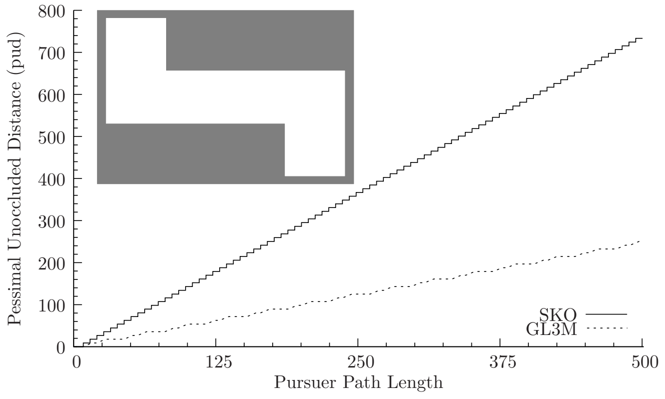

Fig. 8. A simple environment in the shape of a pentomino, inset in a plot of the pessimal unoccluded distance achieved by our algorithm and GL^{3}M, as a function the pursuer's distance traveled in that environment.

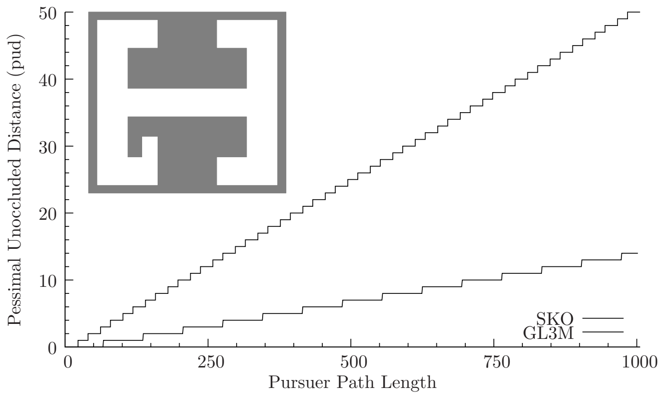

Fig. 9. The "H" environment from [7] inset in a plot of the pessimal unoccluded distance achieved by our algorithm and GL^{3}M, as a function the pursuer's distance traveled in that environment.

There are a number of diverse extensions that could be applied to this work. The planner employed in this work produced feasible pursuer paths capable of driving the unoccluded distance arbitrary high. However, these paths are not optimal in the sense of minimizing the pursuer's travel. The cells of the jump decomposition in which the unoccluded distance changes as a function of the angle formed by the pursuer and an anchor point (as the pursuer rotates about the anchor) are particularly troublesome. Another potential line for future work considers the multirobot formulation of the problem where a planner would coordinate the motions for a team of pursuers. The remaining avenue for future work considers the problem from the evader's perspective rather than the pursuer's. The authors envision a scenario where a deployable robot has a diminishing energy source. Questions that pertain to the evader such as "What strategy could the evader employ to maximize its unoccluded distance while remaining within its energy bound?" are

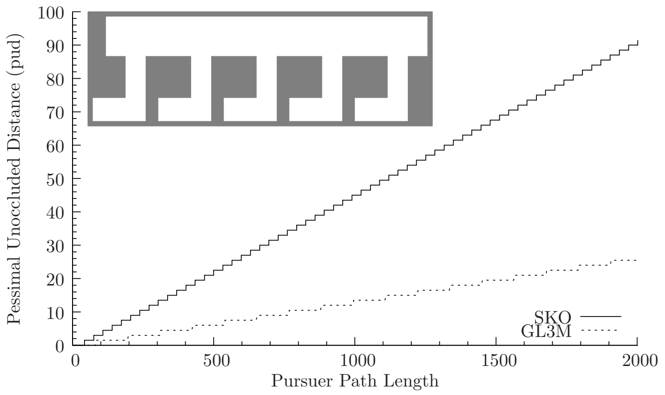

Fig. 10. An environment with some locations that have many simultaneous shadows, inset in a plot of the pessimal unoccluded distance achieved by our algorithm and GL^{3}M, as a function the pursuer's distance traveled in that environment.

of particular interest.

References

[1] S. Bhattacharya, T. Başar, and M. Falcone. Surveillance for security as a pursuit-evasion game. In R. Poovendran and W. Saad, editors, *Decision and Game Theory for Security*, pages 370–379. Springer International Publishing, 2014.

[2] W. Chin and S. Ntafos. Shortest watchman routes in simple polygons. *Discrete and Computational Geometry*, 6(1):9–31, 1991.

[3] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, Berlin, 1997.

[4] J. W. Durham, A. Franchi, and F. Bullo. Distributed pursuit-evasion without mapping or global localization via local frontiers. *Autonomous Robots*, 32(1):81–95, 2012.

[5] B. P. Gerkey, S. Thrun, and G. Gordon. Visibility-based pursuit-evasion with limited field of view. *International Journal of Robotics Research*, 25(4):299–315, 2006.

[6] M. A. Goodrich, B. S. Morse, D. Gerhardt, J. L. Cooper, M. Quigley, J. A. Adams, and C. Humphrey. Supporting wilderness search and rescue using a camera-equipped mini uav: Research articles. *Journal of Field Robotics*, 25(1-2):89–110, January 2008.

[7] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani. Visibility-based pursuit-evasion in a polygonal environment. *International Journal on Computational Geometry and Applications*, 9(5):471–494, 1999.

[8] V. Isler, S. Kannan, and S. Khanna. Randomized pursuit-evasion in a polygonal environment. *IEEE Transactions on Robotics*, 5(21):864–875, 2005.

[9] J. Karelahti, K. Virtanen, and T. Raivio. Near-optimal missile avoidance trajectories via receding horizon control. *Journal of Guidance, Control, and Dynamics*, 30(5):1287–1298, 2007.

[10] B. Kartal, J. Godoy, I. Karamouzas, and S. J. Guy. Stochastic tree search with useful cycles for patrolling problems. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1289–1294, May 2015.

[11] A. Kleiner and A. Kolling. Guaranteed search with large teams of unmanned aerial vehicles. In *Proc. IEEE International Conference on Robotics and Automation*, 2013.

[12] A. Kolling and S. Carpin. Probabilistic graph-clear. In *Proc. IEEE International Conference on Robotics and Automation*, pages 3110–3116, 2009.

[13] A. Kolling and S. Carpin. Surveillance strategies for target detection with sweep lines. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5821–5827, 2009.

[14] X. Lan and M. Schwager. Rapidly-exploring random cycles: Persistent estimation of spatio-temoral fields with multiple sensing robots. *IEEE Transactions on Robotics*, 2016.

[15] S. M. LaValle and J. Hinrichsen. Visibility-based pursuit-evasion: The case of curved environments. *IEEE Transactions on Robotics and Automation*, 17(2):196–201, April 2001.

[16] K. Y. Lin, M. P. Atkinson, T. H. Chung, and K. D. Glazebrook. A graph patrol problem with random attack times. *Operations Research*, 61(3):694–710, 2013.

[17] X. Lin and C. G. Cassandras. An optimal control approach to the multi-agent persistent monitoring problem in two-dimensional spaces. *IEEE Transactions on Automatic Control*, 60(6):1659–1664, June 2015.

[18] N. Michael, E. Stump, and K. Mohta. Persistent surveillance with a team of mavs. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2708–2714, 2011.

[19] R. Murrieta-Cid, T. Muppirala, A. Sarmiento, S. Bhattacharya, and S. Hutchinson. Surveillance strategies for a pursuer with finite sensor range. *International Journal of Robotics Research*, 26(3):233–253, 2007.

[20] T. Ozaslan, S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar. Inspection of penstocks and featureless tunnel-like environments using micro uavs. In *Proc. International Conference on Field and Service Robotics*, 2013.

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

[22] N. N. Petrov. A problem of pursuit in the absence of information on the pursued. *Differentsial'nye Uraveniya (Differential Equations)*, 18:1345–1352, 1982.

[23] M. Pontani and B. A. Conway. Optimal interception of evasive missile warheads: Numerical solution of the differential game. *Journal of Guidance, Control, and Dynamics*, 31(4):1111–1122, 2008.

[24] S. Rajko and S. M. LaValle. A pursuit-evasion bug algorithm. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1954–1960, 2001.

[25] S. Sachs, S. M. LaValle, and S. Rajko. Visibility-based pursuit-evasion in an unknown planar environment. *International Journal of Robotics Research*, 23(1):3–26, 2004.

[26] R. N. Smith, M. Schwager, S. L. Smith, B. H. Jones, D. Rus, and G. S. Sukhatme. Persistent ocean monitoring with underwater gliders: Adapting spatiotemporal sampling resolution. *Journal of Field Robotics*, 28(5):714–741, September 2011.

[27] N. M. Stiffler and J. M. O'Kane. Visibility-based pursuit-evasion with probabilistic evader models. In *Proc. IEEE International Conference on Robotics and Automation*, pages 4254–4259, 2011.

[28] N. M. Stiffler and J. M. O'Kane. Shortest paths for visibility-based pursuit-evasion. In *Proc. IEEE International Conference on Robotics and Automation*, pages 3997–4002, 2012.

[29] N. M. Stiffler and J. M. O'Kane. A complete algorithm for visibility-based pursuit-evasion with multiple pursuers. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1660–1667, 2014.

[30] L. D. Stone, J. O. Royset, and A. R. Washburn. *Optimal search for moving targets*, volume 237 of *International Series in Operations Research and Management Science*. Springer, New York, 2016.

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

[32] P. Tokekar and V. Kumar. Visibility-based persistent monitoring with robot teams. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015.

[33] B. Tovar and S. M. LaValle. Visibility-based pursuit-evasion with bounded speed. In *Proc. Workshop on the Algorithmic Foundations of Robotics*, 2006.

[34] J. Yu, S. Karaman, and D. Rus. Persistent monitoring of events with stochastic arrivals at multiple stations. *IEEE Transactions on Robotics*, 31(3):521–535, 2015.

[35] J. Yu and S. M. LaValle. Shadow information spaces: Combinatorial filters for tracking targets. *IEEE Transactions on Robotics*, 28(2):440–456, 2012.