Planning for robust visibility-based pursuit-evasion

Nicholas M. Stiffler and Jason M. O'Kane

***Abstract*— This paper addresses the problem of planning for visibility-based pursuit evasion, in contexts where the pursuer robot may experience some positioning errors as it moves in search of the evader. Specifically, we consider the case in which a pursuer with an omnidirectional sensor searches a known environment to locate an evader that may move arbitrarily quickly. Known algorithms for this problem are based on decompositions of the environment into regions, followed by a search for a sequence of those regions through which the pursuer should pass. In this paper, we note that these regions can be arbitrarily small, and thus that the movement accuracy required of the pursuer may be arbitrarily high.**

To resolve this limitation, we introduce the notion of an ε-robust solution strategy, in which ε is an upper bound on the positioning error that the pursuer may experience. We establish sufficient conditions under which a solution strategy is ε-robust, and introduce an algorithm that determines, for a given environment, the largest value of ε for which a solution strategy satisfying those sufficient conditions exists. We describe an implementation and show simulated results demonstrating the effectiveness of the approach.

I. Introduction

A number of important applications for autonomous robots can be characterized as *pursuit-evasion problems*. The essence of these tasks is for autonomous robots called *pursuers* to systematically search a domain, possibly in collaboration with humans, to locate one or more other mobile agents called *evaders*. Solutions to such problems have applications in environmental monitoring [1]–[3], surveillance/patrol [4]–[6], disaster recovery [7], and collision avoidance [8]. Substantial effort has been devoted to solving various forms of this problem. However, existing algorithms often rely on very strong assumptions about the pursuers' sensing and movement abilities. Sensors are generally assumed to detect the evaders without fail; actuators are generally assumed to move the robot with extreme precision. Consequently, solutions generated by these algorithms are often brittle or even impossible to execute successfully.

This paper aims to alleviate some of that brittleness for one type of pursuit-evasion problem, the problem of visibility-based pursuit-evasion, in which a pursuer tries to detect an arbitrarily fast evader, using an idealized sensor with omnidirectional view (though that view can still be obstructed by obstacles) and unlimited range.

This material is based upon work supported by the National Science Foundation under Grant No. 1526862. 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

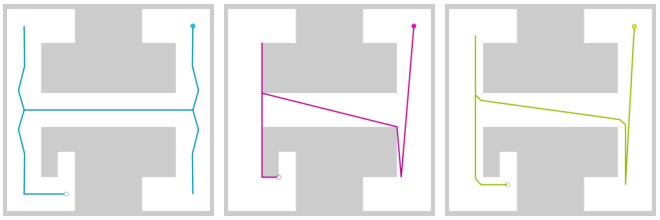


Fig. 1: Solution strategies from three visibility-based pursuit-evasion algorithms in an example environment.

One of the seminal works in visibility-based pursuit-evasion by Guibas, Latombe, LaValle, Lin, and Motwani provides a complete algorithm for this problem [9]. A key contribution of that work that sets it apart from earlier work on similar problems [10] is the visibility cell decomposition that is introduced. The decomposition captures all of the critical information changes as they relate to potential locations for the evaders to be within the environment. This effectively allows for a reformulation of the problem from a continuous search to a discrete graph search on the cells. The authors then introduce an algorithm that utilizes this visibility cell decomposition to produce a path for the pursuer, defined by a sequence of the cells that need to be visited in order, that is guaranteed to locate the evader. We refer to that algorithm, via the initials of the authors' names, as GL³M.

One may observe, however, that GL³M and other more recent approaches that rely on the visibility cell decomposition [11], can generate paths that require the pursuer robot to move through cells that can be arbitrarily small. For example, consider Figure 2, which depicts a conservative region shown in blue that is defined by rays extended from two non-convex environment vertices. If one were to decrease the angle at either of those vertices, the region defined by those rays would become smaller; as that angle approaches zero, the region's size would approach zero as well. If the pursuer robot has some potential for

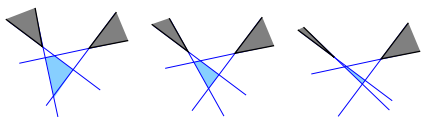


Fig. 2: Visibility cells, as utilized by GL³M, may be arbitrarily small. [left] An example visibility cell. [center, right] Sharper angles at the obstacle vertex generate smaller cells.

inaccuracy in its movements —a real possibility, if not a certainty, where robots are concerned— then that robot may not be able to correctly execute pursuit strategies that pass through such small cells.

The objective of this paper is to generalize the geometric approaches that utilize a visibility cell decomposition of the environment to consider the *robustness* of the generated paths, as measured by the amount of position error those paths can tolerate in the pursuer robot that executes them. Our approach builds upon GL^{3}M because that algorithm remains the definitive complete algorithm for this basic problem. The underlying ideas we introduce, particularly those in Section V, can be generalized to other algorithms based on this sort of decomposition [12]–[15].

After a brief review of related work (Section II), the remainder of the paper is structured as follows:

1) In Section III, we introduce a new form of the classic visibility-based pursuit evasion problem in which the pursuer's movements are subject to bounded deviations of size at most ε from the intended nominal path, and we define the notion of an ε-robust solution strategy that succeeds in locating the evader in spite of these errors.
2) In Section IV, we provide a concise overview of GL^{3}M, focused on mechanisms used there that our algorithm leverages in this new context.
3) In Section V, we lay the foundation for an algorithm that generates ε-robust solution strategies by establishing some sufficient conditions under which a pursuer strategy is ε-robust.
4) In Section VI, we describe an algorithm that determines, for a given environment, the largest value of ε for which those sufficient conditions can be met, and generates a path which achieves that level of robustness.
5) In Section VII, we describe an implementation that illustrates the effectiveness of this approach.

The paper wraps up with some closing discussion in Section VIII.

II. Related Work

Pursuit-Evasion is an active research domain that continues to generate significant interest as a specialization of the broader problem of *search and target tracking*. Pursuit-evasion problems are often characterized by whether the problem is formulated as a differential game [16]–[19], graph [20]–[23], probabilistic [24]–[26], or geometric problem [9], [10], [27].

This paper is concerned with the geometric variant of the problem where the pursuer(s) operate in a geometric environment and attempt to detect the evader(s). There are a number of results for the single pursuer variant of the problem that range from providing theoretical properties, such as completeness [9] and optimality [11], to more restricted scenarios where there are limits on the actuation and sensing capabilities of the pursuers. A sampling of the literature yields the following results which address such limitations: an evader is capable of moving arbitrarily faster than the pursuer [9], bounded velocity for both of the players [28] full visibility for both players [29], limited visibility for both players [30], limited field of view for the pursuer [31], lack of a complete map or localization ability for a pursuer [32], a pursuer that relies on fixed orientations for its sensors [15].

The multi-pursuer variant has received significant interest from the community [33]–[36] not only for broad range of practical applications, but also because of the need for good heuristics owing to the problem complexity [37]. A common thread amongst much of the existing work is an assumption that the pursuer(s) can reliably execute the trajectories generated for them by the planner.

Existing threads of research on safety and robustness in motion planning have focused on accounting for dynamic obstacles [38]–[43] or on sensing uncertainty [40], [42], [43]. This paper addresses a related but distinct problem in which the robustness of the plan is considered in relation to the informative value of the realized trajectory, rather than merely its ability to avoid collisions.

III. Problem Statement

This section formalizes the visibility-based pursuit-evasion problem considered in this paper. First, we describe formal models of the environment, evader, and pursuer (Section III-A), and then we describe the notion of a robust solution to this problem (Section III-B).

A. Representing the Environment, Evader, and Pursuer

A single pursuer moves in search of an evader through a simply connected polygonal region.[1] The environment is a closed and bounded set $W \subset \mathbb{R}^2$, with a polygonal boundary ∂W. Both the evader and pursuer are modeled as points 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. Likewise the pursuer's path, p, is a continuous function $p : [0, T] \to W$, in which T is a finite *stopping time* for the pursuer, and $p(t) \in W$ denotes the position of the pursuer at time $t \in [0, T]$. The function p is called a *motion strategy* for the pursuer. The pursuer moves with a maximum speed, defined without loss of generality to be 1.

It is assumed that both the evader and the pursuer are informed of W but that the evader's trajectory e is unknown to the pursuer.

To detect the evader, the pursuer is equipped with a omnidirectional sensor with unlimited range. The sensor cannot, however, penetrate obstacles. For any point $q \in$

[1] We assume, without loss of generality, that there is exactly one evader. In this setting, any strategy that can guarantee to capture one evader will also capture multiple evaders. If there are no evaders at all, that same strategy can confirm this.

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 within W. Note that $V(q)$ is a closed set.

The time of capture for an evader following trajectory e and a pursuer executing motion strategy p is defined as the earliest time at which the evader's position falls within the pursuer's visibility region:

$$t_c(p, e) = \min \{t \geq 0 \ \mid \ e(t) \in \ V(p(t))\}.$$

Note that, if the pursuer never sees the evader, then $t_c(p, e)$ is undefined.

The pursuer's goal is to capture the evader regardless of the trajectory taken by the evader. The following definition formalizes this idea.

Definition 1: A pursuer motion strategy p is a *solution strategy* if there exists a finite worst-case time of capture over all valid evader trajectories, denoted $t_c(p)$ and defined as

$$t_c(p) = \max_e \ t_c(p, e).$$

That is, for a given pursuer motion strategy p, the time $t_c(p)$ is the least upper bound for the time of capture, considering any possible strategy e for the evader.

B. ε-robust Solution Strategies

We are particularly interested in the case where, due to the usual exigencies of mobile robot actuation, the pursuer may be unable to execute assigned motion strategies exactly. That is, for a given *nominal path* $\widehat{p}$ which the robot intends to follow, the *actual path* p realized by the robot may differ. We model this sort of inaccuracy using a *robustness parameter* ε that quantifies the maximal deviance from a given nominal path. Specifically, we assume that the actual path stays within distance ε of the nominal path. More formally:

Assumption 1: We assume that, though the actual path p executed by the pursuer may differ from the nominal path $\widehat{p}$, the following condition holds: At each time t, the position $p(t)$ achieved by the actual path is within ε of the position $\widehat{p}(t)$ achieved by the nominal path:

$$||p(t) - \widehat{p}(t)|| \leq \varepsilon,$$

for all $t \in [0, T]$.

Figure 3 illustrates this error model.

Our objective is to find motion strategies for the pursuer that are guaranteed to find the evader in spite of this sort of inaccurate execution.

Definition 2: A nominal path $\widehat{p}$ is an *ε-robust solution strategy* if every actual path p satisfying the constraints of Assumption 1 is a solution strategy.

Notice, for example, that every solution strategy is a 0-robust solution strategy. For larger values of ε, having an ε-robust solution strategy is indicative of a strategy that can be successfully executed by a pursuer robot with a lower degree of accuracy in its movements. The remainder of the paper is concerned with algorithms that, for a given environment, generate ε-robust solution strategies for values of ε as large as possible.

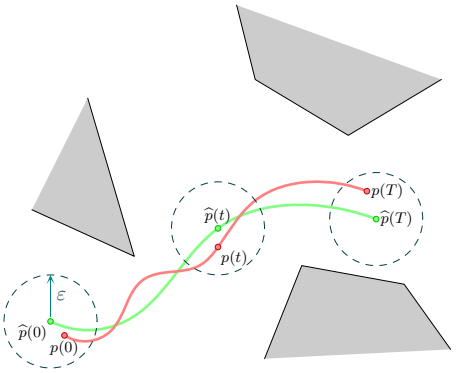


Fig. 3: Parameterized visualizations for a nominal path $\widehat{p}$ and a realized path p. A path p is ε-robust if $\forall\ t$, $0 \leq t \leq T$, the robot following path p remains within ε of the nominal path $\widehat{p}$.

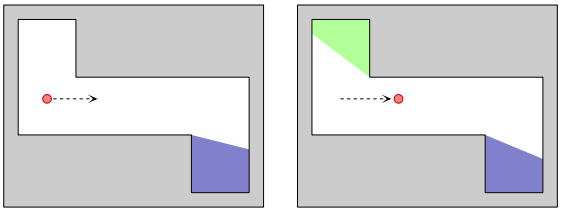


Fig. 4: The shadows can be either *cleared* (green) or *contaminated* (purple) depending on whether an evader can, based on the pursuer's prior movements, be concealed within the shadow. The above shows what would happen if a pursuer (red) travels from the left of the environment to the center of the environment.

IV. Review of GL^{3}M

This section reviews the approach used by GL^{3}M to solve this pursuit-evasion problem in general, without regard to robustness. We leverage and adapt these concepts for our new ε-robust planning algorithm.

If the evader is not visible to the pursuer, then it must be concealed in one of the finitely many regions of the environment that are occluded from view. We call these regions *shadows* (Figure 4). Though the shadows themselves change continuously as the pursuer moves through the environment, the cardinality of the shadow set changes only when a *shadow event* occurs, i.e. a shadow *appears*, *disappears*, *splits*, or *merges* with another shadow.

Because the evader can move as quickly as it likes within each shadow, the crucial piece of information about each shadow for our pursuit-evasion problem is just a single bit indicating whether or not the evader might be hiding in that shadow. A shadow s is called *cleared* at time t if, based on the pursuer's motions up to time t, it is not possible for the evader to be within s without having been captured. A shadow is called

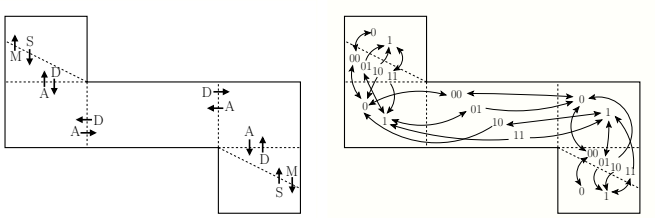


Fig. 5: (left) Events describe how the shadows will change when the pursuer crosses a conservative region boundary. The visibility events are (A)ppear, (D)isappear, (S)plit, and (M)erge.(right) The complete Pursuit-Evasion Graph is a directed graph with shadow labels that update when the pursuer crosses a cell boundary.

contaminated if it is not clear. That is, a contaminated shadow is one in which the evader may be hiding. It will be convenient to assign binary labels to each shadow, so that a label of 0 means the shadow is cleared and a label of 1 means the shadow is contaminated.

The shadow events induce the following changes to the clear/contaminated status of a shadow:

- Appear: A new shadow can appear, when a previously visible part of the environment becomes hidden — assign 0.
- Disappear: An existing shadow can disappear, when a pursuer moves to a location from which that region is visible — discard the label.
- Split: A shadow can split into multiple shadows, when the pursuers move in such a way that a given shadow is no longer path-connected — copy original label to all new shadows.
- Merge: Multiple existing shadows can merge into a single shadow, when previously disconnected shadows become path-connected — 0 if and only if the merging shadows were also 0, and 1 otherwise.

The basis of the GL^{3}M algorithm —and of our extension to it— is that these labels fully capture the relevant information about the pursuer's progress in searching for the evader. That is, a motion strategy $p : [0, T] \to W$ is a solution strategy if and only if, at time T, all of the shadows are clear. This result is useful because it confirms that, using this worst-case reasoning, we can completely represent the pursuer's progress in searching for the evader by its current configuration and the current shadow labels.

The GL^{3}M algorithm employs a visibility cell decomposition of the environment corresponding to the shadow events that can occur, which discretizes the environment into a collection of conservative regions, i.e. regions of the environment in which the shadow cardinality and labels remain the same provided the pursuer does not induce another shadow event (Figure 5–left).

Based on this analysis of the changes to the shadow labels, GL^{3}M computes a solution strategy (without regard for robustness) by forming and searching a graph, which we call the *pursuit-evasion graph (PEG)*. An example of the PEG can be seen in Figure 5–right. Each node in the PEG represents a region in the visibility cell decomposition, coupled with a set of specific binary labels for the shadows that exist in that cell. Directed edges in the PEG connect nodes corresponding to adjacent cells, with appropriate updates to the shadow labels as described above.

V. Sufficient conditions for ε-robust strategies

This section proves a sufficient condition for a solution strategy to be ε-robust. Notice that if $\widehat{p}$ is a solution, and p visits the same sequence of conservative regions as $\widehat{p}$, then p is a solution as well. Unfortunately, if $\varepsilon > 0$, we cannot guarantee that this condition holds: any time $\widehat{p}$ passes within distance ε of a conservative region boundary, p has an opportunity to experience a shadow event not experienced by $\widehat{p}$. Note that this is true even when $\widehat{p}$ approaches a conservative region boundary that it intends to cross. Our goal is to identify conditions for $\widehat{p}$ under which p is nonetheless still a solution.

Recall from Section IV that the shadow labels change when the pursuer's motion induces a shadow event $\Lambda \in \{A(ppear), D(isappear), S(plit), M(erge)\}$. Each visibility event λ has a complementary visibility event, denoted λ^{-1} that is triggered if the pursuer were to cross from the opposite direction. In general,

- Appear events A are complementary to Disappear events D, and vice versa.
- Split events S are complementary to Merge events M, and vice versa.

This leads to the following result.

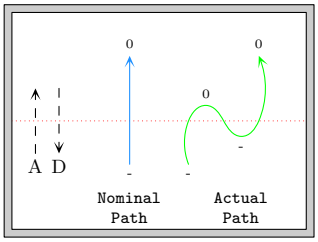
Lemma 1: If the path p is a solution strategy, where $\lambda_1 \cdots \lambda_k \lambda_{k+1} \cdots \lambda_K$ is the sequence of shadow events induced by p, then any path p' which generates a event sequence of the form $\lambda_1 \cdots \lambda_k \lambda_k^{-1} \lambda_k \lambda_{k+1} \cdots \lambda_K$ —that is, with $\lambda_k^{-1}\lambda_k$ inserted after λ_k— is also a solution strategy.

Proof: By case analysis. For the backtracking transition $\lambda_k \lambda_k^{-1} \lambda_k$, we must consider the event type of λ_k, along with the clear/contaminated labels of any shadows that are affected by the event λ_k. Thus 8 distinct cases arise:

(a) if λ_k is an appear event,
(b) if λ_k is a disappear event for a clear shadow,
(c) if λ_k is a disappear event for a contaminated shadow,
(d) if λ_k is a split event for a clear shadow,
(e) if λ_k is a split event for a contaminated shadow,
(f) if λ_k is a merge event for two clear shadows,
(g) if λ_k is a merge event for one clear and one contaminated shadow, and
(h) if λ_k is a merge event for two contaminated shadows.

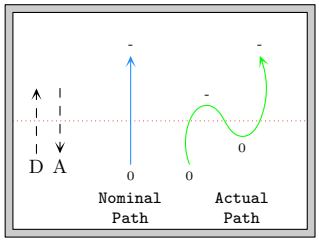
For each of these cases, we verify that a backtracking of this nature does not affect the state of the search. Details for each case appear in Figure 6. ■

Notice that repeated applications of Lemma 1 can account for actual paths that introduce multiple instances of this sort of backtracking, as illustrated in Figure 7. To formalize this idea, we first delineate the class of paths



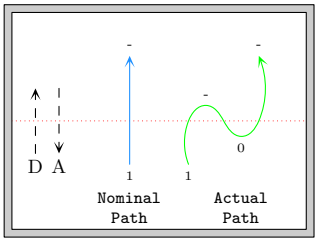
A $AA^{-1}A$

(a) When a single A event occurs, an empty label gets assigned a 0, this is the same as a sequence of the form $AA^{-1}A$.



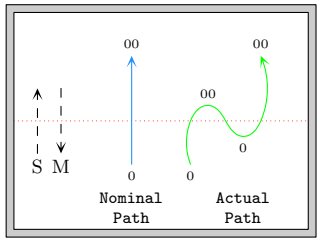
D $DD^{-1}D$

(b) When a single D event occurs for a 0 label, the label is dropped (empty). This is the same as a sequence of the form $DD^{-1}D$.



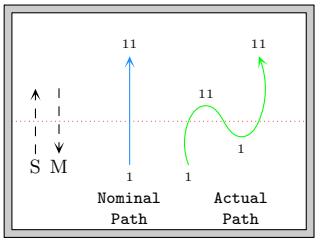
D $DD^{-1}D$

(c) When a single D event occurs for a 1 label, the label is dropped (empty). This is the same as a sequence of the form $DD^{-1}D$.



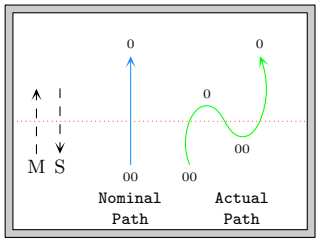
S $SS^{-1}S$

(d) When a single S event occurs for a 0 label, the shadow splits resulting in two 00 labels. This is the same as a sequence of the form $SS^{-1}S$.



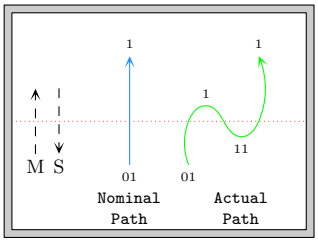
S $SS^{-1}S$

(e) When a single S event occurs for a 1 label, the shadow splits resulting in two 11 labels. This is the same as a sequence of the form $SS^{-1}S$.



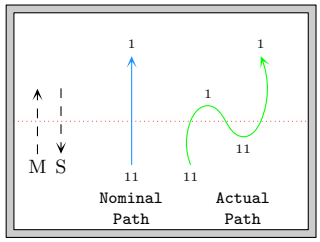
M $MM^{-1}M$

(f) When a single M event occurs for a 00 label, the shadows merge into a single 0 label. This is the same as a sequence of the form $MM^{-1}M$.



M $MM^{-1}M$

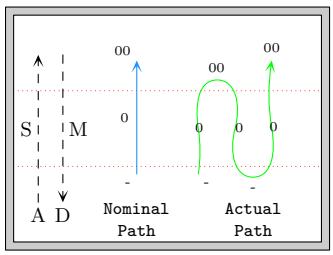
(g) When a single M event occurs for a 01 or 10 label, the shadows merge into a single 1 label. This is the same as a sequence of the form $MM^{-1}M$.



M $MM^{-1}M$

(h) When a single M event occurs for a 11 label, the shadows merge into a single 1 label. This is the same as a sequence of the form $MM^{-1}M$.

Fig. 6: Cases for the proof of Lemma 1.



AS $ASS^{-1}A^{-1}AS$

Fig. 7: Beginning with an empty label, first there is an A event that produces a 0 label that then S into a 00 label. This is equivalent to the sequence $ASS^{-1}A^{-1}AS$. Note that there are multiple events that are backtracked. First a split, $\cancel{ASS^{-1}}A^{-1}AS = AA^{-1}AS$, and then an appear, $\cancel{AA^{-1}}AS = AS$.

for which this sort of temporary backtracking is the only change in the event sequence that can occur.

Definition 3: A nominal path $\widehat{p}$ that induces a conservative region sequence $r_1, \ldots, r_m$ is called *ε-region-sequence-preserving (ε-RSP)* if, for each r_i, the portion of $\widehat{p}$ within r_i travels

- no closer than ε to any vertex of r_i, and
- no closer than ε to any boundary segment of r_i except the boundary segments shared between r_i and r_{i-1} and between r_i and r_{i+1}.

The intuition is that an ε-RSP path stays at least ε distance away from the conservative region boundaries, except those that are crossed by $\widehat{p}$. We can now show that ε-RSP is indeed a sufficient condition for ε-robust.

Theorem 1: If $\widehat{p}$ is ε-RSP, then $\widehat{p}$ is also ε-robust.

Proof: Consider a continuous deformation f from $\widehat{p}$ to p, under which $f(0) = \widehat{p}$, $f(1) = p$, and for each $\alpha \in [0, 1]$, $f(\alpha)$ is a possible actual path under error bound ε for $\widehat{p}$. Let $\Lambda(\alpha)$ represent the sequence of shadow events experienced by the path $f(\alpha)$. Notice that, because f is continuous, Λ is a piecewise-constant function. Moreover, changes in Λ occur at values of α where the path adds or eliminates a single-step backtracking pair. Lemma 1 ensures that, if the path before each such change is a solution, then the resulting path after the change is also a solution. Thus, by induction on the number of changes, conclude that p is a solution. ■

As a consequence, the algorithm in the next section ensures that the paths it generates are indeed ε-RSP, for as large an ε as possible.

VI. Algorithm Description

This section presents an algorithm that computes an ε-RSP pursuer strategy through an environment, for as large an ε as possible. We begin by describing a

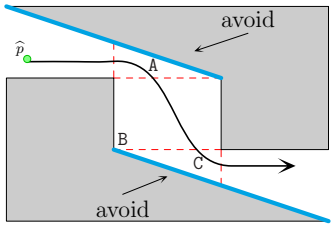


Fig. 8: Given a region sequence R = ABC, notice that the most robust nominal path through region B is influenced by curves belonging to A and C. Specifically, the path through B should turn to avoid the diagonal blue segments (which are avoid segments). That is, the path through B is impacted by the existence of other regions, both before B and after it.

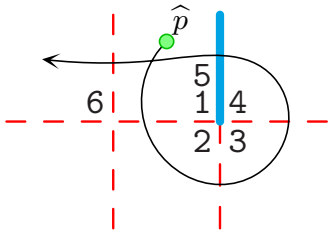


Fig. 9: Given R = 123456, when computing k_1, k_2 for $r_i = 2$, notice that $k_1 = 1$ and $k_2 = 3$ is maximal because of the blue boundary which is an avoid curve from k_1 which necessitates that $\widehat{p}$ cross it in region 4.

subroutine which, given a starting point p and a sequence of conservative regions $\mathtt{R} = (r_1, \ldots, r_n)$, computes a path through those regions. We then outline a complete algorithm that utilizes the aforementioned subroutine to explore progressively larger sequences of conservative regions, culminating in the pursuer either clearing the environment, or determining that no such sequence exists.

A. Computing ε-robust Subpaths

First, we consider the problem of generating a subpath through a single conservative region $r_i \in \mathtt{R}$ in a given sequence. The challenge is that the most robust nominal path through conservative region r_i may, and often times will, be affected not just by the geometry of r_i, but also by that of other regions in the sequence. Figure 8 illustrates such a scenario. As a result, we cannot simply generate paths along the medial axis of each cell in isolation.

Instead, we must identify the boundary curves and boundary vertices that may impact $\widehat{p}$ as it traverses through each region r_i. Given this collection of elements, we can generate the appropriate subpath through each r_i by computing and walking along the medial axis induced by this entire element set.

To determine the relevant curves and vertices that affect the subpath through r_i, which we term avoid elements, we begin by including all of the vertices of r_i along with the boundary curves of r_i except for the curve from which the robot enters from r_{i-1} and the curve from which the robot exits to r_{i+1}.

Then we identify indices k_1 and k_2, with $k_1 < i < k_2$, for which every boundary segment in $r_{k_1}, \ldots, r_{k_2}$ is either a curve that $\widehat{p}$ must cross or a curve that $\widehat{p}$ must avoid. That is, we walk alternatingly forward (i.e. increasing k_2) and backward (decreasing k_1) until reaching values of k_1 and k_2 for which, if either were extended any further, a curve would be added that must be avoided by one in one step but crossed in another step. Figure 9 illustrates the process.

After finding appropriate values for k_1 and k_2, we extract the segments from $r_{k_1}, \ldots, r_{k_2}$ that must be avoided for the path through r_i to satisfy the ε-RSP conditions, and compute their medial axis. We then walk through this graph, from its (necessarily unique, because the regions are convex) entry point to r_i from r_{i-1} to its exit into r_{i+1}. Note that a special case occurs when $r_{i-1} = r_{i+1}$, wherein the entrance and exit curves are the same. In this case, we instead select a path along the medial axis graph from the combined entrance/exit curve, to the interior point of r_i at the center of its largest enclosed circle; the robot's path would enter r_i, travel to this center point, and then depart along the same path.

Finally, the full path visiting regions $r_1, \ldots, r_n$ is then recoverable by concatenating all of the subwalks for each individual r_i. For each each subwalk, we compute its nearest approach to any of its avoid segments, denoted ε_i. By construction, the full walk is then $(\min_i \varepsilon_i)$-RSP and thus $(\min_i \varepsilon_i)$-robust. Once this bound is found, the length of full path can be optimized by 'cutting corners' to reduce the clearance across the path down to this lower bound.

B. Forward Search

Finally, the overall planner at the top level performs a forward search over sequences of conservative regions. The search algorithm maintains a priority queue of search nodes. Each search node $\mathtt{S}_i$ contains the following information:

- PEG node sequence $\mathtt{N}(\mathtt{S}_i) = n_1 \ldots n_k$ denoted as $\mathtt{N}_i$.
- A path $\widehat{p}$ through $\mathtt{N}_i$, denoted as $\widehat{p}_i$.
- An ε for which $\widehat{p}_i$ is ε-RSP.

The priority queue is seeded with a search node consisting of a region sequence containing the initial region, an empty path, and the corresponding ε. The search nodes in the queue are ordered according to the ε-robustness of $\widehat{p}$; in the case of a tie, the secondary criterion is path length, with shorter length preferred.

At each iteration, the search extracts the highest-priority node $\mathtt{S}_i$ from the queue, constructs new search nodes $\mathtt{S}'$ by appending a new adjacent conservative region to the end of $\mathtt{N}_i$ and computing (using the approach from Section VI-A) the corresponding $\widehat{p}'$ and ε', and inserting this new search node $\mathtt{S}' = (\mathtt{N}', \widehat{p}', \varepsilon')$ into the queue.

To accelerate the process, prior to insertion into the queue, we apply a pruning operation that determines whether the search node can be safely discarded. The pruning strategies are very similar to those that appear

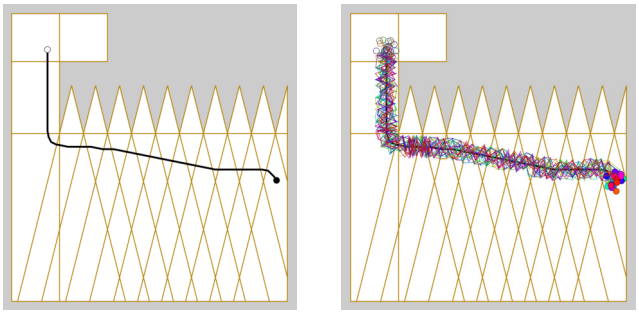


Fig. 10: (left) An ε-robust nominal path (black) through an environment. (right) A collection of actual paths that remain within ε of the nominal path(black).

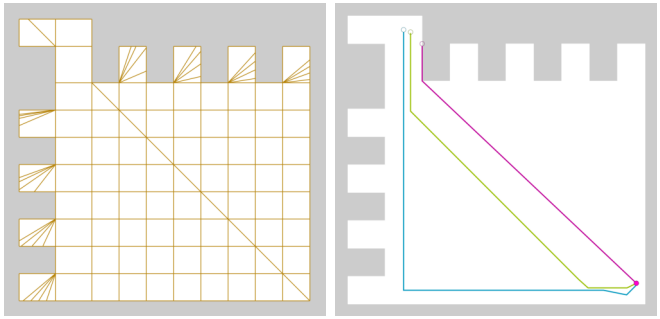


Fig. 11: Scenario where the ε-robust (green), GL^{3}M (blue), and Optimal (magenta) strategies traverse through different regions.

in our prior work [11]. We employ the CYCLE-FREE and REGRESSION pruning strategies which for a search node $\mathtt{S}_i$ ensures that a PEG sequence $\mathtt{N}_i$ is cycle free and that there is no regression, in terms of discernible labels, during the search. Additionally, we provide the following strategy to prune provably suboptimal nodes: Given two search nodes $\mathtt{S}_j$ and $\mathtt{S}_k$: If $\mathtt{R}_j = (R_1, \ldots, r_a, r_b)$, $\mathtt{R}_k = (r_1, \ldots, r_y, r_z)$ where $1 \leq a < b$, $1 \leq y < z$ have $r_a = r_y$ and $r_b = r_z$, and $p_b^j = p_z^k$ **AND** either

- $\varepsilon(\mathtt{S}_j) > \varepsilon(\mathtt{S}_k)$ **or**
- $\varepsilon(\mathtt{S}_j) = \varepsilon(\mathtt{S}_k)$ and $\textsc{dist}(\mathtt{S}_j) < \textsc{dist}(\mathtt{S}_k)$.

Then $\mathtt{S}_k$ can be pruned. Intuitively, if two search nodes culminate with the pursuer generating the same visibility event and they share the same shadow label if one can provably reach that area with a lower epsilon or perhaps with the same epsilon but faster, then any future strategy that builds on the suboptimal strategy with forever be suboptimal had the search chosen to just expand the other search node. Thus it can be discarded.

The search terminates when either a search node $\mathtt{S}_i$ reaches the top of the queue whose path $\widehat{p}_i$ is a solution strategy (success), or the queue becomes empty (no solution exists).

VII. Case Studies

This section presents some results for an implementation of the algorithm described in Section VI. There are two primary characteristics that we were interested in investigating: a) How robust is the computed plan to actuation error, namely what kinds of actual paths are realizable as solution strategies when given an ε-robust nominal path, and b) How does an ε-robust solution strategy compare to the complete and optimal strategies?

Figure 10 presents a scenario where the actual path deviates from the nominal path by no more than ε and serves as an illustration for the kind of error tolerance that the ε-robust strategy compensates for. Note that the actual path contains several intermediary (event–event complement) pairs that do not affect the result.

Due to the search prioritizing clearance (ε-robustness) before optimizing the path, it is possible for the nominal path $\widehat{p}$ to visit a sequence of conservative regions that differs from the complete algorithm and the optimal algorithm as seen in Figure 11. In scenarios where the sequence of conservative regions is the same, the ε-robust strategy behaves very similarly to the optimal solution. Note however that the optimal solution is, by construction, only 0-robust and is particularly prone to errors when there is a sudden change in direction (See Figure 1). These are accounted for in the ε-robust strategy.

VIII. Conclusion

This paper presents a measure of the robustness of a pursuer motion strategy for a class of visibility-based pursuit-evasion problems that accounts for motion uncertainty by the pursuer. A set of sufficient conditions is provided, and these conditions are leveraged into an algorithm for computing a solution strategy that maximizes this clearance measure.

The most direct avenue for future research is an extension of this work, which considers only *sufficient* conditions for ε-robustness, into a study on the *necessary* conditions. The current measure is restricted to deformations that maintain an equivalent event sequence, up to some possible temporary backtracking. However, it may be possible for the actual path to deviate from the nominal path more severely while remaining a solution strategy — some events impact the correctness of the solution, others do not. We would seek to understand how these kinds of adjacently permissible interactions would affect the overall robustness of a pursuer motion strategy.

This work specifically considers a holonomic drive pursuer. An investigation into how the idea of ε-robustness can be adapted to non-holonomic systems is another avenue for future work.

References

[1] M. Dunbabin and L. Marques, "Robots for environmental monitoring: Significant advancements and applications," *IEEE Robotics and Automation Magazine*, vol. 19, pp. 24–39, 2012.

[2] V. Isler, N. Noori, P. Plonski, A. Renzaglia, P. Tokekar, and J. V. Hook, "Finding and tracking targets in the wild: Algorithms and field deployments," in *Proc. IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2015.

[3] P. Tokekar, D. Bhadauria, A. Studenski, and V. Isler, "A robotic system for monitoring carp in Minnesota lakes," *Journal of Field Robotics*, vol. 27, no. 3, pp. 681–685, 2010.

[4] J. J. Acevedo, B. C. Arrue, I. Maza, and A. Ollero, "A decentralized algorithm for area surveillance missions using a team of aerial robots with different sensing capabilities," in *Proc. IEEE International Conference on Robotics and Automation*, 2014.

[5] P. Dames, P. Tokekar, and V. Kumar, "Detecting, localizing, and tracking an unknown number of moving targets using a team of mobile robots," *International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1540–1553, 2017.

[6] T. Alam, M. M. Rahman, L. Bobadilla, and B. Rapp, "Multi-vehicle patrolling with limited visibility and communication constraints," in *Military Communications Conference*, 2017, pp. 465–479.

[7] G. Hollinger, A. Kehagias, and S. Singh, "Probabilistic strategies for pursuit in cluttered environments with multiple robots," in *Proc. IEEE International Conference on Robotics and Automation*, 2007.

[8] V. Isler, D. Sun, and S. Sastry, "Roadmap based pursuit-evasion and collision avoidance," in *Proc. Robotics: Science and Systems*, 2005.

[9] 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*, vol. 9, no. 5, pp. 471–494, 1999.

[10] I. Suzuki and M. Yamashita, "Searching for a mobile intruder in a polygonal region," *SIAM Journal on Computing*, vol. 21, no. 5, pp. 863–888, Oct. 1992.

[11] N. M. Stiffler and J. M. O'Kane, "Complete and optimal visibility-based pursuit-evasion," *International Journal of Robotics Research*, vol. 36, no. 8, pp. 923–946, July 2017.

[12] G. Dudek, K. Romanik, and S. Whitesides, "Localizing a robot with minimum travel," *SIAM Journal on Computing*, vol. 27, no. 2, pp. 583–604, 1998.

[13] 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*, 2011, pp. 4254–4259.

[14] N. M. Stiffler, A. Kolling, and J. M. O'Kane, "Persistent pursuit-evasion: The case of the preoccupied pursuer," in *Proc. IEEE International Conference on Robotics and Automation*, 2017, pp. 5027–5034.

[15] N. M. Stiffler and J. M. O'Kane, "Pursuit-evasion with fixed beams," in *Proc. IEEE International Conference on Robotics and Automation*, 2016, pp. 4251–4258.

[16] R. Isaacs, *Differential Games*. New York: Wiley, 1965.

[17] Y. C. Ho, A. Bryson, and S. Baron, "Differential games and optimal pursuit-evasion strategies," *IEEE Transactions on Automatic Control*, vol. 10, no. 4, pp. 385–389, Oct. 1965.

[18] R. Zou and S. Bhattacharya, "On optimal pursuit trajectories for visibility-based target tracking game," *IEEE Transactions on Robotics*, vol. 35, pp. 449–465, Apr. 2019.

[19] A. V. Moll, D. Casbeer, E. Garcia, and D. Milutinovic, "Pursuit-evasion of an evader by multiple pursuers," in *Proc. International Conference on Unmanned Aircraft Systems*, June 2018, pp. 133–142.

[20] T. D. Parsons, "Pursuit-evasion in a graph," in *Theory and Application of Graphs*, Y. Alavi and D. R. Lick, Eds. Berlin: Springer-Verlag, 1976, pp. 426–441.

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

[22] A. Kolling and S. Carpin, "Pursuit-evasion on trees by robot teams," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 32–47, 2010.

[23] R. Borie, S. Koenig, and C. Tovey, "Pursuit-evasion problems," in *Handbook of Graph Theory*, J. Gross, J. Yellen, and P. Zhang, Eds. Chapman and Hall, 2013, ch. 9.5, pp. 1145–1165.

[24] J. Hespanha, M. Prandini, and S. Sastry, "Probabilistic pursuit-evasion games: A one-step nash approach," in *Proc. IEEE Conference on Decision and Control*, vol. 3, Feb. 2000, pp. 2272–2277.

[25] R. Vidal, O. Shakernia, H. J. Kim, D. H. Shim, and S. Sastry, "Probabilistic pursuit-evasion games: Theory, implementation, and experimental evaluation," *IEEE Transactions on Robotics and Automation*, 2002.

[26] F. Shkurti, N. Kakodkar, and G. Dudek, "Model-based probabilistic pursuit via inverse reinforcement learning," in *Proc. IEEE International Conference on Robotics and Automation*. IEEE, 2018, pp. 7804–7811.

[27] S. Park, J. Lee, and K. Chwa, "Visibility-based pursuit-evasion in a polygonal region by a searcher," in *Proc. International Colloquium on Automata, Languages and Programming*. Springer-Verlag, 2001, pp. 281–290.

[28] B. Tovar and S. M. LaValle, "Visibility-based pursuit-evasion with bounded speed," *International Journal of Robotics Research*, vol. 27, pp. 1350–1360, 2008.

[29] D. Bhadauria, K. Klein, V. Isler, and S. Suri, "Capturing an evader in polygonal environments with obstacles: The full visibility case," *International Journal of Robotics Research*, 2012.

[30] A. Q. Li, F. Amigoni, R. Fioratto, and V. Isler, "A search-based approach to solve pursuit-evasion games with limited visibility in polygonal environments," in *Proc. International Conference on Autonomous Agents and Multiagent Systems*, 2018, pp. 1693–1701.

[31] B. P. Gerkey, S. Thrun, and G. Gordon, "Visibility-based pursuit-evasion with limited field of view." *International Journal of Robotics Research*, vol. 25, no. 4, pp. 299–315, 2006.

[32] S. Sachs, S. M. LaValle, and S. Rajko, "Visibility-based pursuit-evasion in an unknown planar environment," *International Journal of Robotics Research*, vol. 23, no. 1, pp. 3–26, 2004.

[33] A. Kolling and S. Carpin, "Multi-robot pursuit-evasion without maps," in *Proc. IEEE International Conference on Robotics and Automation*, 2010, pp. 3045–3051.

[34] F. B. Joseph W. Durham, Antonio Franchi, "Distributed pursuit-evasion without mapping or global localization via local frontiers," *Autonomous Robots*, vol. 32, pp. 81–95, 2012.

[35] L. Gregorin, S. Givigi, E. Freire, E. Carvalho, and L. Molina, "Heuristics for the multi-robot worst-case pursuit-evasion problem," *IEEE Access*, vol. 5, pp. 17 552–17 566, Aug. 2017.

[36] N. M. Stiffler and J. M. O'Kane, "A sampling based algorithm for multi-robot visibility-based pursuit-evasion," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 1782–1789.

[37] 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*, 2014, pp. 1660–1667.

[38] S. Petti and T. Fraichard, "Safe motion planning in dynamic environments," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 2210–2215.

[39] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns," *Autonomous Robots*, vol. 35, no. 1, pp. 51–76, 2013.

[40] S. Patil, J. Van Den Berg, and R. Alterovitz, "Estimating probability of collision for safe motion planning under gaussian motion and sensing uncertainty," in *Proc. IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 3238–3244.

[41] O. Gal, Z. Shiller, and E. Rimon, "Efficient and safe on-line motion planning in dynamic environments," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 88–93.

[42] C. Fulgenzi, A. Spalanzani, and C. Laugier, "Probabilistic motion planning among moving obstacles following typical motion patterns," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 4027–4033.

[43] N. E. Du Toit and J. W. Burdick, "Robot motion planning in dynamic, uncertain environments," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 101–115, 2011.