2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Evader Surveillance under Incomplete Information

Israel Becerra[†], Rafael Murrieta-Cid[†] and Raul Monroy[‡]
[†]Centro de Investigación en Matemáticas, CIMAT
Guanajuato México
{israelb, murrieta}@cimat.mx
[‡]Tecnológico de Monterrey
Campus Edo. de México
raulm@itesm.mx

***Abstract*— This paper is concerned with determining whether a mobile robot, called the *pursuer*, is up to maintaining visibility of an antagonist agent, called the *evader*. This problem, a variant of *pursuit-evasion*, has been largely studied, following a systematic treatment by increasingly relaxing a number of restrictions. In [8], we considered a scenario where the pursuer and the evader move at bound speed, traveling around a known, 2D environment, which contains obstacles. Then, considering that, in an attempt to escape, the evader travels the shortest path to reach a potential escape region, we provided a decision procedure that determines whether or not the pursuer is up to maintain visibility of the evader and obtained complexity measures of this surveillance task.**

In this paper, we prove that there are cases for which an evader may escape only if it does not travel the shortest path to an escapable region. We introduce planning strategies for the movement of the pursuer that keeps track of the evader, even if the evader chooses not to travel the shortest path to an escape region. We also present a sufficient condition for the evader to escape that does not depend on the initial positions of the players. It can be verified only using the environment. All our algorithms have been implemented and we show simulation results.

I. Introduction

This paper is to do with pursuit-evasion. We are concerned with determining whether a mobile robot, the *pursuer*, is up to maintaining visibility of an antagonist agent, the *evader*.

In a previous paper, [8], we considered a scenario where the pursuer and the evader move at bound speed, traveling around a known, 2D environment, which contain obstacles. To attempt to escape, the evader was assumed to travel the *shortest path to escape*. In this paper, we take a step further: we provide motion planning strategies for a pursuer which has to keep track of an evader which may choose *not* to follow *the shortest path to escape policy*. This new setting is of interest, because, as will be shown in this paper, there exist evasion paths that require the evader *not* to travel the shortest path to escape. Unlike [8], the pursuer does not know about the global paths to be taken by the evader but knows where it will be after a small progress of time.

A. Contributions

In this paper, we make four main contributions: (1) We show that, regardless of the relative speed of each player, if the pursuer does not know the evader motion policy then *there exist* cases where the evader can escape *only if* it does not follow the shortest path to escape policy. (2) We show that determining whether or not a pursuer can maintain visibility of the evader at all times depends on two general factors: (i) the initial positions of both the pursuer and the evader; and (ii) the long-term path plans that can be executed by the evader. (3) We present motion planning algorithms that enable the pursuer to keep track of an evader who may not follow the shortest path to escape policy. (4) We present a sufficient condition for the evader to escape that does not depend on the initial positions of the players and which can be verified by looking at the environment only. Our algorithms have been implemented and simulation results are shown.

B. Related Work

Keeping track of a moving evader is a popular, long-standing problem which has been studied from several perspectives. For example, [7] approached it applying game theory. The result is an algorithm which attempts to maximize the probability that the evader will remain visible in the future. [3] suggested a method which, unlike [7], does not use a global map of the environment; instead, using a local map, built with the help of a range sensor, [3] applies a combinatorial algorithm that computes the motion for the pursuer. [5] approached evader surveillance using a greedy approach. To drive the greedy motion planning algorithm, [5] applied a local minimum risk function, called the vantage time. More related to ours is the work of [4], which shows how to efficiently compute a pursuer optimal path in response to a given evader movement in a known environment. In [10] a related problem is presented, the authors seek conditions under which the evaders can prevent being seen by the pursuer. Pursuit-evasion has found to be use in interesting applications. For example, in [6], the authors noticed the similarity between pursuit-evasion games and mobile-routing for networking. Applying this similarity, they proposed motion planning algorithms for robotic routers to maintain connectivity between a mobile user and a base station. A preliminary version of this work is reported in [9].

II. Problem definition

The evader and the pursuer are modeled as points moving over a known environment. The environment contains obstacles, each of which is modeled as a polygon. Every participant is assumed to accurately know its position at all times, is equipped with an omni-directional sensor, and it is limited to move at bound speed. Other than these, no kinematic nor dynamic constraints are imposed on the pursuer or the evader.

The pursuer does not know the motion policy of the evader, who moves continuously and antagonistically. The pursuer is not able to predict the evader motion policy or to learn it. However, it is assumed to know where the evader will be after a small progress of time, Δt. So, we assume a universal clock which ticks every Δt units of time; clock ticks are then used to index periods of time. The pursuer is thus able to know the whereabouts of the evader, from t to $t + \Delta t$.

Under this setting, we address the problem of discovering pursuer motion strategies that are able to maintain strong mutual visibility of the evader, considering that the global motion policy of it is unknown. As will see in Section II-A, strong mutual visibility is stronger than classical visibility. Similarly, it is our goal to seek for sufficient conditions, independent of the initial position of the players, such that they imply that the evader is bound to escape.

A. Strong Mutual Visibility

Let $R_1, \ldots, R_n$ be a partition of the environment, $W = \bigcup_i R_i$, such that each R_i ($i \in \{1, \ldots, n\}$) is a convex region. The evader is under pursuer surveillance if strong mutual visibility of the evader by the pursuer holds [8]. Two regions are *strongly mutually visible* if every point belonging to any of the two regions is able to see all the points of the other region. The pursuer maintains strong mutual visibility of the evader, if it is within the same region where the evader is or if they both are in regions that are strongly mutually visible. Thus, maintaining strong mutual visibility of the evader amounts to maintaining visibility of the entire region where it is.

III. Preliminaries

Region convexity ensures that a robot with omnidirectional sensing is able to see all the points within the region of residence. Our convex partition is similar to the region decomposition produced by the lines of the aspect graph in 2D using perspective projection [2], plus an additional feature, namely: every pair of bitagent vertices are connected. In our partition, bi-tangent rays are extended outward and inward from a pair of bi-tangent points [8]. The partition of the environment yields two graphs, one called *Accessibility Graph* (*AG*) and the other *Mutual Visibility Graph* (*MVG*). In each graph, nodes represent regions. In an AG, two nodes R_i and R_j are connected, written $(R_i, R_j) \in AG$, if their associated regions share a region boundary bigger than one single point. Likewise, in an MVG, two nodes R_i and R_j are connected, written $(R_i, R_j) \in MVG$, if their associated regions are strongly mutually visible. Using the MVG and its current position, each participant is able to know both which regions are candidates to attempt to escape, called *escapable regions*, and which regions the pursuer should move to if an escape is to be prevented, called *prevention-from-escape regions*.

An MVG therefore provides information to find a sufficient condition to maintain evader visibility while an AG provides possible region transitions that either participant can carry out. Note that what counts as an escapable (respectively prevention-from-escape) region depends on the current regions where both the evader and the pursuer are. More precisely, let E_i (respectively P_j) denote that the evader (respectively the pursuer) is at region R_i (respectively R_j). For each pair $\langle E_i, P_j \rangle$, denoting a problem configuration, the set of escapable regions, written $\mathcal{R}^e_{(i,j)} \subseteq \text{int}(W)$, is given by $\{R : (R_j, R) \notin MVG\}$. Moreover, for every escapable region $R \in \mathcal{R}^e_{(i,j)}$, there is a set of prevention-from-escape regions, written $\mathcal{R}^p_{(i,j)}(R) \subseteq \text{int}(W)$, given by $\{R' : (R', R) \in MVG\}$.

A. Bound Speed

Given a problem configuration, $\langle E_i, P_j \rangle$, the primary constraint governing pursuit-evasion is given as a relation on two times: the time taken for the evader to reach an escapable region, $t_e(R_{e\langle i,j \rangle})$, for some $R_{e\langle i,j \rangle} \in \mathcal{R}^e_{(i,j)}$, and the time taken for the pursuer to reach one associated prevention-from-escape region, $t_{pe}(R_{pe}(R_{e\langle i,j \rangle}))$, for some $R_{pe}(R_{e\langle i,j \rangle}) \in \mathcal{R}^p_{(i,j)}(R_{e\langle i,j \rangle})$.

For the pursuer to prevent the evader from escaping, the constraint $t_e(R_{e\langle i,j \rangle}) \geq t_{pe}(R_{pe}(R_{e\langle i,j \rangle}))$ must be satisfied at all times, for all $R_{e\langle i,j \rangle} \in \mathcal{R}^e_{(i,j)}$. Considering that both pursuer and evader travel a given path, possibly at a different speed, this constraint can be defined in terms of distances and relative velocities:

$$d_e(E(e), R_{e\langle i,j \rangle}) \geq d_{pe}(P(pe), R_{pe}(R_{e\langle i,j \rangle})) \frac{V_e}{V_{pe}} \qquad (1)$$

where V_e and V_{pe} are respectively the speed of the evader and the pursuer, and $E(e)$ and $P(pe)$ are the positions of the evader and the pursuer. It is worth noticing that d_e and d_{pe} are, in general, *geodesic distances*. This formulation *holds for polygons with or without holes*. However, in polygons with holes a faster evader can always escape pursuer surveillance following a simple strategy: turn around the nearest hole [8].

IV. The effect of Incomplete Information over the Paths to Escape

In [8], we have shown that traveling the shortest-path to reach an escapable region is the best policy for the evader if the pursuer knows which escapable region the evader is aiming to.

We now show that shortest path is not the best escape policy in the more general case, where the pursuer does not know which region, among a collection, the evader will choose to go in an attempt to escape. In fact, we will show that there exist cases for which evasion is plausible only under the proviso that the evader does not adopt the shortest path to escape policy.

The careful reader will have already noticed that our result takes the form:

$$\exists x.(\mathit{MayEscape}(E, x) \land \neg \mathit{Follows}(E, \mathit{ShortestPath})) \qquad (2)$$

where x ranges over configurations and E stands for the evader. Theorem (2) accounts for a counterexample to the appealing conjecture:

$$\forall x.(\mathit{MayEscape}(E,x) \rightarrow \mathit{Follows}(E, \mathit{ShortestPath})) \quad (3)$$

which happens *not* to be a theorem.[1] Clearly, there are infinitely too many configurations for which the shortest path policy enables the evader to escape. Producing any one such an instance is trivial even for a primary school pupil. Even though all these success cases, (3) is not a theorem.

Our non-trivial result, (2), holds regardless of whether the evader is slower or faster than the pursuer. Let us consider first the case of a faster evader.

Proposition 4.1: There exist cases where a faster evader can escape only it does not travel the shortest distance from its initial position to a escapable region.

Proof: Fig. 1 depicts the scenario in which we elaborate our proof. There, E stands for the evader, P for pursuer. Let $A(p)$ denote that player A is at distinguished point $p \in \Re^2$.

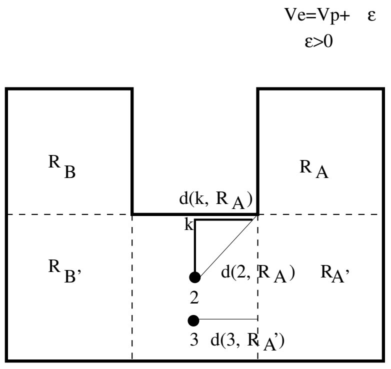

Fig. 1. Evader faster than the pursuer

For the initial system configuration, $(E(2), P(3))$, there are two escapable regions, R_A and R_B, each of which has two prevention from escape regions, $\{R_A, R_{A'}\}$ and $\{R_B, R_{B'}\}$, respectively. Given that strong mutual visibility holds, then if the evader, traveling the shortest path distance, goes to either R_A or R_B, the pursuer is able to prevent escape correspondingly going to either the nearest point that belongs to $R_{A'}$ or $R_{B'}$. $d(E(2), R_A) > d(P(3), R_{A'})\frac{Ve}{Vp}$ and $d(E(2), R_B) > d(P(3), R_{B'})\frac{Ve}{Vp}$. Notice that the pursuer always goes to the nearest prevention from escape region; this explains why going to R_A or R_B is not considered as an option.

Now notice that if the evader first goes to point k, then it will simultaneously diminish the distance to both escapable regions. We emphasize that moving this way the evader is not traveling the shortest path to any of either escapable region (indeed, along this way it is not even moving toward an escapable region). But notice that the pursuer cannot achieve a similar goal: move to a place where the distance to both prevention from escape regions, $R_{A'}$ and $R_{B'}$ simultaneously diminishes. Once at k, the evader has a wining move, given that it is faster than the pursuer. This is because $d(E(k), R_A) = d(P(3), R_{A'})$ and $d(E(k), R_B) = d(P(3), R_{B'})$. It follows, that the evader can escape only when it does not travel the shortest path to escape from its initial position. ■

The rationale behind this escape is that the pursuer does not know where the evader is heading at in a long term and so he has to take into account all possible escape regions. We now consider the second case, where the evader is slower than the pursuer.

Proposition 4.2: There exist scenarios for which a slower evader can escape only if it does not travel the shortest distance from its initial position to a escapable region.

Proof: Refer to Fig. 2.

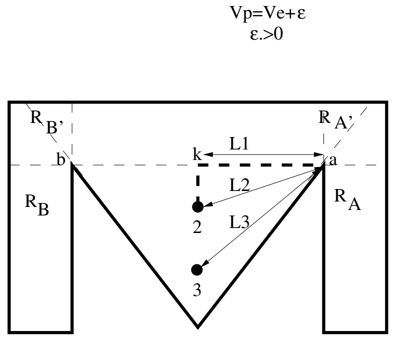

Fig. 2. Evader slower than the pursuer

At first, the evader is at position $E(2)$ and the pursuer at $P(3)$, and thus the system configuration is $(E(2), P(3))$. Let a and b respectively be the nearest point both to escapable regions R_A and R_B. Notice that in this case these points also belong to $\{R_A, R_{A'}\}$ and $\{R_B, R'_B\}$, the associated prevent from escape regions and they are also the nearest points to prevent escape. Let $L_1 = d(k,a)$, $L_2 = d(2,a)$ and $L_3 = d(3,a)$ and assume both that $L_1 < L_2 < L_3$ and that $\frac{L_1}{L_2} < \frac{L_2}{L_3}$.[2] Without loss of generality, assume that both players move at saturated speed and that $V_p = \frac{L_3}{L_2}V_e$. Then $d(3,2) > d(2,k)$. Moreover, assume that the time that the evader needs to travel L_2, denoted, $t_e(L_2)$, equals the time the pursuer needs to travel L_3, denoted $t_p(L_3)$.

First, notice that, under these conditions, if the evader attempted to reach a traveling L_2, the pursuer would be able to catch up, traveling L_3. However, if the evader went to k, the pursuer would attempt to move to a place that simultaneously reduces the distance that separates it from both a and b; that is, to point 2.[3] But then the pursuer is bound to fail. This is because in the new system configuration $(E(k), P(2))$, for the pursuer to catch up the evader it would need to travel at $V_p = \frac{L_2}{L_1}V_e$, but this contradicts our initial assumption, namely: $\frac{L_1}{L_2} < \frac{L_2}{L_3}$. To see this, notice that to catch up in the first step $V_p = \frac{L_3}{L_2}V_e$, but in the second step $V_p = \frac{L_2}{L_1}V_e$ but this implies that the velocity of the pursuer must be bigger than the bound $\frac{L_3}{L_2}$. ■

[1] Indeed, we arrived at (2) in an attempt to prove (3).

[2] The interested reader will have noticed that this condition is essential for the argument to follow.

[3] In what follows, we omit from our reasoning the prevention of a escape onto b, but recall that the pursuer must deal with both escape points at once.

Thus, together, propositions 4.1 and 4.2, show that (3) is indeed *not* a theorem: there exists cases where an evader can escape only if it does not take the shortest path to escape, one of the key contributions of this paper.

V. Keeping or Escaping Pursuer Surveillance

Any solution to the problem of determining whether or not the pursuer is able to maintain evader surveillance on a given environment depends on two main factors: (i) the initial position of both participants; and (ii) the long-term combinatoric paths that the evader can travel over the environment in an attempt to escape. We will study both factors below. We formulate our problem as a game and so, for every match, we will determine which among the pursuer or the evader has a winning strategy.

A. Initial Conditions

Finding a winner to an instance of our game depends clearly on the initial position of both players, as well as their corresponding maximum speed. There might be configurations that either player would find unpleasant. Consider, for instance, the case where, even though strong-mutual visibility holds, the players are so apart one another, that, to escape, the evader may just need to go to the adjacent region. To see a concrete example of this case, refer to Fig. 2. If the pursuer and the evader respectively are at point a and b and if neither player is faster than the other, then the evader will be in no troubles at all to escape.

Given an instance of the problem, to determine whether or not there is still a game, we proceed as follows. First, use the MVG and the AG, together with (1), to find out whether there is a escapable region that the evader can reach in a time strictly smaller than that needed by the pursuer to reach a corresponding prevent-from-escape region. If there does not exist any one such a escapable region, the game continues; otherwise, there is a winning path for the evader.

Our method performs similarly to that of [1], even though the latter method considers classical visibility[4]. This is because, in this case, building a compact set is analogous to identifying whether the evader is able to reach a escape region before the pursuer prevents the escape.

B. Combinatoric Paths

Determining which player has a winning strategy *also* depends on graphs that capture fundamental aspects of the environment. We have found that such graphs can be computed when considering the best paths to escape and the sets of elements in $\Re^2$ that are up to prevent the escape even whenever any one such best escape path is taken. We call any one such set of elements in $\Re^2$ *an Ω-border*, since the elements are all placed in the border between two regions in the MVG. The evader has a winning strategy, if there is a path for which an associated Ω-border collapses to the empty set. Ω-borders are computed without considering initial conditions; rather, the analysis is performed in a steady-state condition, where only the path determines the size and shape of each Ω-border.

To compute every Ω-border, we made the evader to travel every single shortest-time path starting on a reflex vertex[5] and visiting any other reflex vertex. These paths are all in the *reduced visibility graph* [11]. We are interested only in paths of the reducibility graph because any one time the evader reaches a reflex vertex a new possibility for an escape comes up. This is in turn because every reflex vertex, by definition, breaks convexity of the environment.

The rationale behind the algorithm below is to find out whether the pursuer can keep surveillance (respectively, the evader can escape) at a long-term, assuming valid initial conditions. The evader travels the reduced visibility graph, choosing a visit ordering which aims to make the time to escape smaller than the time to prevent escape. Notice that this involves dealing with an intractable problem [8].

Below, we present an algorithm which plans pursuer motions so as to keep track of an evader who does not necessarily travel the shortest paths to an escapable region. This algorithm consists of two methods. The first method uses the network of shortest distance between borders of escapable regions in order to define valid points for the pursuer departure. These points, which depend on the velocity of both players, form the Ω borders. The second method uses Ω borders in order to identify regions where the pursuer should go to upon each move of the evader.

It is important to underline that while the Ω borders are computed assuming that the evader travels moving in the network of shortest path between reflex vertices, the Ω borders are used to prevent the escaping of the evader even if it does not move traveling those shortest paths.

We use Ω borders to compute a region in the plane where the pursuer must be in order to prevent the evader from escaping. We call this region S, for solution set. $S \in \Re^2$ is a set of points, which guarantee that at a given instant of time, the evader cannot reach a reflex vertex in a time strictly smaller than the time that the pursuer needs to reach an Ω border.

1) Ω borders: Before presenting the algorithm to compute the Ω borders, we need to define some preliminary concepts.

Let v_k be a reflex vertex in the polygonal workspace W. Inciding in v_k, there are two segments of W. Conversely, emerging from v_k there are two inflection rays of the aspect graph [2]. For each vertex v_k, we order its associated inflection rays using a counter-clockwise ordering. Thus, the first inflection ray of v_k on this ordering is called r_{k1} and the second one r_{k2}.

Each inflection ray is a potential Ω-border, which is to be refined by our algorithm below. Ω-border refinement depends on the paths that the evader travels over the reduced visibility graph [11]. The evader might turn around any vertex either clockwise or counter-clockwise. If the evader turns around counter-clockwise a vertex v_k, then it crosses the r_{k1} inflection ray first; otherwise, it crosses r_{k2} first.

[4]In classical visibility, two points see one another if the line segment between them does not cross an obstacle

[5]A *reflex vertex* is one of an internal angle greater than π.

Let $\mathcal{V}$ be the set of all reflex vertices in W. Then, we have to analyze all permutational paths, each of which is of the form $p_k = v_{k1} \rightarrow v_{k2} \rightarrow \cdots \rightarrow v_{k|\mathcal{V}|} \rightarrow v_{k1}$ if it is a cycle or $p_k = v_{k1} \rightarrow v_{k2} \rightarrow \cdots \rightarrow v_{k|\mathcal{V}|}$ if it is a sequence (note that in the worst case p_k has a size of $|\mathcal{V}| + 1$ or $|\mathcal{V}|$ respectively, but it can be shorter). In particular, the inflection rays to be considered between two consecutive reflex vertices on the path depend on which direction the evader turns around the vertices: clockwise or counter-clockwise. Basically, this analysis determines the inflection ray to be considered next. For the ordering $v_{ki} \rightarrow v_{ki+1}$, the only possible options are $r_{ki+1,1}$ or $r_{ki+1,2}$. The selection process is as shown in table I.

v_{ki}	v_{ki+1}	r_{ki}	r_{ki+1}
$\circlearrowright$	$\circlearrowright$	$r_{ki,2}$	$r_{ki+1,2}$
$\circlearrowright$	$\circlearrowleft$	$r_{ki,2}$	$r_{ki+1,1}$
$\circlearrowleft$	$\circlearrowleft$	$r_{ki,1}$	$r_{ki+1,1}$
$\circlearrowleft$	$\circlearrowright$	$r_{ki,1}$	$r_{ki+1,2}$

TABLE I

INFLECTION RAYS TO BE USED AS Ω-BORDERS

Up to this point, we have a set Ω_k for each p_k, which contains a collection of potential Ω borders (the inflection rays) that the pursuer must visit as the evader goes through each reflex vertex in the order specified by a given path p_k. In order to prevent an escape, the pursuer would need to reach the Ω border Ω_{ki} at least at the same time that the evader reaches the reflex vertex v_{ki}. What the algorithm shown below does, it is to refine these potential Ω borders taking into account the interaction between them (the refinement is done by taking pairs of potential Ω borders). For example, supposing that at the beginning the Ω borders are the respective inflection rays given by a path p_k and that the pursuer starts from Ω_{k1}, it may happen that in a first step the pursuer will only be able to reach on time some portion of the initial Ω_{k2}, so when we calculate the reachable portion of Ω_{k3} in a second step, we will be restricted to take as an start set only the portion of Ω_{k2} that we were able to reach at the first step; this chain of restrictions must be carried up to the final Ω border defined by the path.

Our algorithm for computing the Ω borders, see Algorithm 1, takes into account two cases, one when p_k is a sequence and second, when p_k is a cycle. When p_k is a sequence we initialize each Ω_{ki} in Ω_k with its respective r_{ki} based on p_k and table I. For the refining process we take pairs $(\Omega_{ki},\ \Omega_{ki+1})$ starting from $i = 1$ to $|\mathcal{V}| - 1$, where we generate a reachable area around Ω_{ki} and then we intersect it with Ω_{ki+1} to get the reachable portion of Ω_{ki+1} which will be reused to perform the same calculations for the next pair $(\Omega_{ki+1},\ \Omega_{ki+2})$. After going through all pairs, we obtain a refined $\Omega_{k|\mathcal{V}|}$ that is reachable from Ω_{k1}. The next step is to go backwards through pairs $(\Omega_{ki},\ \Omega_{ki-1})$ with $i = |\mathcal{V}|$ to 2, to recover the valid portions of each Ω_{ki} that gave us the reachable portion of $\Omega_{k|\mathcal{V}|}$ from Ω_{k1}. This is done by generating a reachable area around Ω_{ki} and then intersecting it with Ω_{ki-1} to get the correct portion of Ω_{ki-1}. We iteratively do the same procedure for all pairs. At the end we obtain a valid Ω_{k1} that is a subset of r_{k1} which allow us to reach a valid subset of $r_{k|\mathcal{V}|}$ that is $\Omega_{k|\mathcal{V}|}$.

Algorithm 1 Computing Ω borders

```
Input: Work space, $W$, environment partition.
Output: $\Omega$ borders.
for every permutational path of the form
        $p_k = v_{k1} \rightarrow v_{k2} \rightarrow \cdots \rightarrow v_{k|\mathcal{V}|} \rightarrow v_{k1}$
or
        $p_k = v_{k1} \rightarrow v_{k2} \rightarrow \cdots \rightarrow v_{k|\mathcal{V}|}$
 do
  1. Considering the pair $v_{ki} \rightarrow v_{ki+1}$, look up at Table I
  to determine the inflection rays: $r_{ki}$ and $r_{ki+1}$;
  2. Initialization. $\Omega_k = \{\Omega_{ki} : i \in \mathbb{N},\ 1 \le i \le n\}$, where
  $n = |\mathcal{V}|$ for a sequence or $n = |\mathcal{V}| + 1$ for a cycle;
  for $i = 1$ to $|\mathcal{V}|$ do
    A. $\Omega_{ki} \leftarrow r_{ki}$;
  end for
  if $p_k$ is a cycle then
    B. $\Omega_{k|\mathcal{V}|+1} \leftarrow \Omega_{k1}$;
  end if
    C. $bd \leftarrow true$;
  3. $\Omega_k$ calculation;
  if $p_k$ is a cycle then
    repeat
      A. $\Omega_{k|\mathcal{V}|+1} \leftarrow \Omega_{k1}$;
      B. $\Omega_k \leftarrow \Omega BdrsInteraction(\Omega_k,\ p_k,\ |\mathcal{V}|{+}1, bd)$;
      C. $bd \leftarrow false$;
    until $\Omega_{k|\mathcal{V}|+1} \subseteq \Omega_{k1} \lor \Omega_{k|\mathcal{V}|+1} = \phi$
  else if $p_k$ is a sequence then
    A. $\Omega_k \leftarrow \Omega BdrsInteraction(\Omega_k,\ p_k,\ |\mathcal{V}|, bd)$;
  end if
  4. Store($\Omega_k$, $p_k$);
end for
```

When p_k is a cycle the procedure is basically the same as the one for sequences. The main difference is that the last Ω border $\Omega_{k|\mathcal{V}|+1}$ in Ω_k refers to the same inflection ray as Ω_{k1}. We perform the same refining procedure described above to get a valid Ω_{k1} to go to a valid $\Omega_{k|\mathcal{V}|+1}$, but recalling that both Ω_{k1} and $\Omega_{k|\mathcal{V}|+1}$ are subsets of r_{k1} we have to do some extra analysis here. After refining the potential Ω borders and supposing that the evader would take this path, three things can occur. First, if $\Omega_{k|\mathcal{V}|+1}$ is an empty set, this means that the pursuer can't maintain visibility of the evader (the evader wins the game). Second, if $\Omega_{k|\mathcal{V}|+1} \subseteq \Omega_{k1}$, this means that the pursuer would be able to maintain visibility of the evader and he would be able to keep maintaining it if the evader decides to take this path again and again. Third, if $\Omega_{k1} \subseteq \Omega_{k|\mathcal{V}|+1}$, this means that the pursuer could maintain visibility of the evader but once they have completed a cycle, the pursuer could have ended in a non valid starting position, in which case, we repeat the refinement procedure again until we find ourselves in the first or second case described above. Notice that since the length of the winning pursuer paths are lower bounded by tours over the reduced visibility graph (the shortest evader path), those tours provide a convergence condition for the algorithm for computing the Ω borders.

Algorithm 2 $\Omega BdrsInteraction(\Omega_k, p_k, n, bd)$

```
Input: Set $\Omega_k$ composed by unrefined $\Omega$ borders related to $p_k$, path $p_k$, $n$ length of $p_k$, $bd$ flag that indicates that the calculation must be done in both directions;
Output: Set $\Omega_k$ with refined $\Omega$ borders due to the interaction between them.
if $bd$ == true then
  for $i = 1$ to $n - 1$ do
    1. $\Omega_{ki+1} \leftarrow Refine(\Omega_{ki}, \Omega_{ki+1}, v_{ki}, v_{ki+1})$;
  end for
end if
for $i = n$ to 2 do
    2. $\Omega_{ki-1} \leftarrow Refine(\Omega_{ki}, \Omega_{ki-1}, v_{ki}, v_{ki-1})$;
end for
Return $\Omega_k$;
```

Algorithm 3 Refine Ω borders, $Refine(\Omega_{ka}, \Omega_{kb}, v_{ka}, v_{kb})$

```
Input: $\Omega$ border $\Omega_{ka}$, $\Omega$ border $\Omega_{kb}$, reflex vertex $v_{ka}$ and reflex vertex $v_{kb}$.
Output: refined $\Omega$ border.
1. $A = \{x \in \mathbb{R}^2 : d(p, x) \leq d(v_{ka}, v_{kb})\frac{V_{pe}}{V_e}, p \in \mathbb{R}^2, p \in \Omega_{ka}\}$;
Return $A \cap \Omega_{kb}$;
```

Consider Fig. 3. In both parts, the environment is the polygon shown with black solid lines, the region partition is shown with dashed lines and the regions are labeled with numbers. The polygon has 4 reflex vertices. The Ω borders, computed setting $Ve = Vp$, are shown in dark gray (green) color. In Fig. 3 A) the Ω borders were computed using the vertices $\{a, b, c\}$; here, the resulting Ω borders are two points and one line segment. In Fig. 3 B) the Ω borders were computed using all the reflex vertices $\{a, b, c, d\}$; now the resulting Ω borders are simply points. Notice that when the vertex d is considered, the Ω borders are reduced to the vertices themselves. This is because the border of the partition regions and the vertices are equally separated one another. Notice that in this environment, if we set $V_e > V_p$, then the Ω borders would be empty sets. Thus, in this environment, a faster evader will always win.

In general, computing Ω borders requires dealing with a computationally intractable problem. However, we can find an approximate solution, for example, by computing Ω borders for a subset of reflex vertices. This would be useful if we clustered the vertices, hence dividing the tour and then compute Ω borders for each part of this tour. This is equivalent to do local planning; the planning horizon would be determined by the number of vertices to consider. This strategy will not guarantee surveillance at all times but it

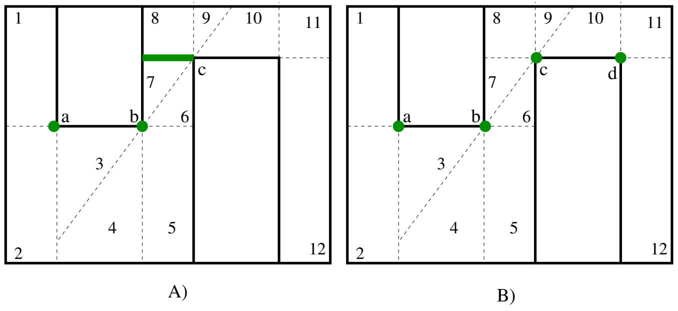

Fig. 3. Ω borders

is useful to make short term planning that prevent evader escaping. Of course, for small polygons, e.g. with around 15 reflex vertices or so, it is actually possible, using a regular PC, to compute the Ω borders for all reflex vertices.

C. Regions of Local Solution

S is the set of points where the pursuer must be to prevent the evader from going behind a reflex vertex, v_k, and hence escape. Let $\mathbf{V}$ be a subset of all reflex vertices that fulfill the properties described in Proposition 5.1. Then, considering that the evader and the pursuer respectively are at region R_i and R_j, S is defined by (4).

Proposition 5.1: V is a subset of all the reflex vertices which have the next two properties: first, they must share a point with some region not mutually visible with R_j (region where the pursuer is located) and second, they must be visible from the evader's position.

Proof: First of all, all the the vertices that are part of a region mutually visible with R_j are not candidates to be taken as part of V because they do not produce an escape to the evader. The region related to them is currently visible to the pursuer. By contrast, due to the condition that the escapes must take place going through a reflex vertex and supposing that the evader is at a given position, if the evader wants to reach a vertex that is not visible to him from that position, he will need to pass through a reflex vertex that he can see from the given position. All the posible escapes through the vertices behind a vertex that is visible to the evader's position were already considered in the construction of the Ω borders related to it, hence the only thing left to do is to make sure that the pursuer reaches the correct Ω border before the evader reaches the visible reflex vertex. This is done by placing the pursuer inside the S set. ■

In equation (4), $\Omega(v_k)$ denotes the Ω border associated to reflex vertex v_k, $d(p, \Omega(v_k))$ the *geodesic distance* between the point p and $\Omega(v_k)$, $d(P(e), v_k)$ denotes also a *geodesic distance*, this time between the evader position $P(e)$ and vertex v_k, and $\mathbf{V}$ the subset of all the reflex vertices, as described in proposition 5.1 (that is, we consider only the reflex vertices that share a point with some R not strong mutually visible to R_j that are visible from the current evader's position.)

The region S is used to define a new valid pursuer position regardless the trajectory of the evader. Thus, we have a

$$S = \{p \in R \;\;:\;\; (R, R_i) \in MVG) \bigwedge_{v_k \in \mathbf{V}} d(p, \Omega(v_k)) \leq d(P(e), v_k)\frac{V_{pe}}{V_e})\} \tag{4}$$

method to compute the pursuer motion, which is independent of the evader policy and path.

VI. Players Strategies and Paths

We have found that under the definition of strong mutual visibility, the possible paths that the evader can travel to escape can be classified in two types: 1) paths where the evader escapes when it does not touch a reflex vertex in the environment 2) paths where the evader escapes but the opposite condition holds.

The first types of paths do not lie on the visibility graph. Fig. 4 shows a path of type 1). As before, the environment is the polygon shown with back solid lines, the region partition is shown with dashed lines and the regions are labeled with numbers. The evader is at region 1 and the pursuer at region 21. If the evader goes to region 2, then the pursuer must go to region 12 (the closest prevention from escape region from the current pursuer position).

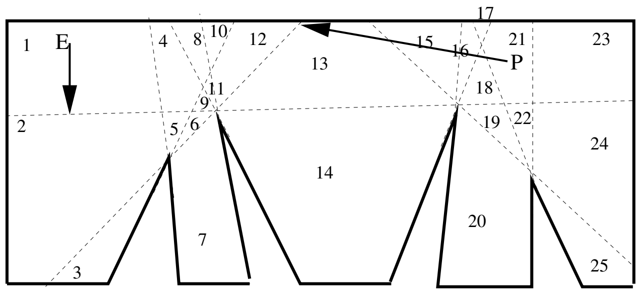

Fig. 4. Paths type 1

Arrows represent paths. P stands for the pursuer and E for the evader. These paths cannot be characterized based only on the reflex vertices positions. But notice that Equation (1) can be used to determine whether or not at a given time the evader can escape. At all instants of time, based on the position of the players, together with the MVG and the GV, it is possible to decide whether or not the evader has a winning move.

For the evader to travel the second type of paths, it may move along the reduced visibility graph. The motivation for the evader to do so is whenever there is an empty Ω border, it will eventually win (indeed when it reaches the associated vertex). Notice that this condition is independent of the initial position of the players and can be determined using only the map. This is a sufficient condition for the evader to win.

VII. Simulation Results

All our simulation experiments were run on a dual core processor PC, equipped with 2 GB of RAM, running Linux. Fig. 6 shows snapshots of our simulations. In these figures, obstacles are shown in gray and the free space in white, regions are delimited with line segments. The evader position is the (red) square and the pursuer is the (blue) circle. The Ω borders are shown in dark gray (green) and the S regions in light gray (yellow).

In the simulation the velocity of the evader was set equal to the velocity of the pursuer $Ve = Vp$. In Fig. 6, the evader does not travel the shortest paths to escape. It is interesting to see that when the pursuer gets close to the edges of the reduced visibility graph, the associated S set becomes smaller. In Fig. 6 c), when the evader touches the obstacle (it is on an edge of the reduced visibility graph), the S region collapses to a single point (the pursuer must be at the same position where the evader is). Finally, notice that the S regions are delimited either by line segments or arcs of circles. Note that the Ω borders used to compute the S set will change depending on the regions in which both the evader and the pursuer reside, and the S set itself depends on the current position of the players. Recall that all Ω borders for all possible paths (sequence of reflex vertices) and regions in which the players may reside are pre-computed and stored, when the game takes place the appropriate Ω borders are retrieved. For instance, if the pursuer and the evader are in the region shown in Fig. 6 b) then the corresponding Ω borders are points and line segments, but if the players are in the regions shown in Fig. 6 e) then the Ω borders are only points.

The running time of our software for computing the Ω borders of the environment shown in Fig. 6 was 40 ms. The S regions were computed in approximatively 33 ms. These very small running times are due to the small number of reflex vertices (only 4) for this environment.

Fig. 5 shows in dark gray (green) the Ω borders corresponding to the tour of vertices $1 \to 2 \to 3 \to 4 \to 5 \to 1$ (see figure to locate the corresponding vertices). These Ω borders were computed in 45 ms. Since the environment is small, in this scenario we compute the Ω borders with an evader tour that considers all the reflex vertices. We are currently testing our programs in bigger environments (50 reflex vertices or so).

VIII. Conclusion

In this paper we have proved that if the pursuer does not know the evader motion policy then there are cases where an evader can escape only if it does not travel the shortest distance from its initial position to a escapable region, regardless whether the evader is faster or slower than the pursuer. We have presented an algorithm which plans pursuer motions so as to keep track of an evader who does not necessarily travel the shortest paths to an escapable region. We have found a sufficient condition for the evader to escape that does not depend on the initial positions of the players. It only depends on the environment. Therefore, this condition can be checked, before the game

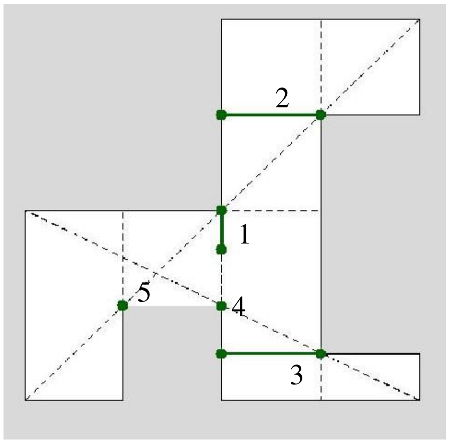

Fig. 5. Simulation Results

starts, if the condition holds then there is no motivation to play. Finally, we have implemented all our algorithms and presented simulation results.

REFERENCES

[1] S. Bhattacharya and S. Hutchinson, Approximation Schemes for two-players pursuit evasion games with visibility constraints, *In Proc Int Conf. Robotics Science and Systems IV*, 2008.

[2] K. W. Bowyer and C. R. Dyer, Aspect graphs: An introduction and survey of recent results, *Int. J. Imaging Syst. Technol.*, vol 2, pp. 315-328, 1990.

[3] H.H. Gonzalez, C.-Y. Lee and J.-C. Latombe, Real-Time Combinatorial Tracking of a Evader Moving Unpredictably Among Obstacles, *In Proc IEEE Int. Conf. on Robotics and Automation*, 2002.

[4] A. Efrat, H. H. Gonzalez, S. G. Kobourov, and L. Palaniappan. Optimal Motion Strategies to Track and Capture a Predictable Target. *Proc. IEEE Int. Conf. on Robotics and Automation*, Taipei, Taiwan, pp. 411-423, 2003.

[5] T. Bandyopadhyay, Y. Li, M.H. Ang and D. Hsu, A Greedy Strategy for Tracking a Locally Predictable Target among Obstacles, *In Proc IEEE Int. Conf. on Robotics and Automation*, 2006.

[6] O. Tekdas and V. Isler, Robotic Routers. *In Proc IEEE Int. Conf. on Robotics and Automation*, 2008.

[7] S.M. LaValle, H.H. Gonzalez, C. Becker and J.-C. Latombe, Motion Strategies for Maintaining Visibility of a Moving Target. *In Proc IEEE Int. Conf. on Robotics and Automation*, 1997.

[8] R. Murrieta-Cid, R. Monroy, S. Hutchinson and J. P. Laumond A Complexity Result for the Pursuit-Evasion Game of Maintaining Visibility of a Moving Evader, *IEEE International Conference on Robotics and Automation 2008*, pp 2657-2664.

[9] M. Santes, R. Murrieta-Cid and R. Monroy, Evader Surveillance under Incomplete Information, *Technical Report CIMAT No I-09-03/24-02-2009*.

[10] B. Tovar and S. M. LaValle, Visibility-based Pursuit-Evasion with Bounded Speed, Proc. Workshop on the Algorithmic Foundations of Robotics (WAFR), 2006.

[11] J. O'Rourke, *Computational Geometry In C*. Cambridge University Press, 2000.

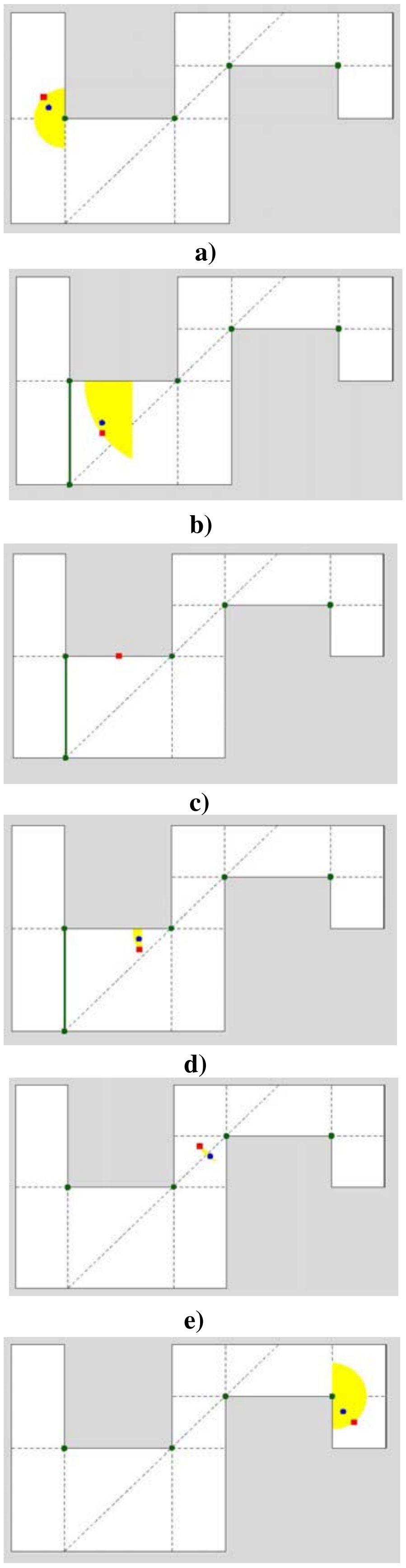

Fig. 6. Simulation Results