Auton Robot (2016) 40:395–423
DOI 10.1007/s10514-015-9477-5

Maintaining strong mutual visibility of an evader moving over the reduced visibility graph

Israel Becerra[1] · **Rafael Murrieta-Cid**[1] · **Raul Monroy**[2] · **Seth Hutchinson**[3] · **Jean-Paul Laumond**[4]

Received: 31 October 2013 / Accepted: 18 July 2015 / Published online: 2 August 2015

Abstract In this paper, we address the problem of determining whether a mobile robot, called the *pursuer*, is able to maintain strong mutual visibility (a visibility notion between regions over a convex partition of the environment) of an antagonist agent, called the *evader*. We frame the problem as a non cooperative game. We consider the case in which the pursuer and the evader move at bounded speed, traveling in a known polygonal environment with or without holes, and in which there are no restrictions as to the distance that might separate the agents. Unlike our previous efforts (Murrieta-Cid et al. in Int J Robot Res 26:233–253, 2007), we give special attention to the combinatorial problem that arises when searching for a solution through visiting several locations in an environment with obstacles. In this paper we take a step further, namely, we assume an antagonistic evader who moves continuously and unpredictably, but with a constraint over its set of admissible motion policies, as the evader moves in the shortest-path roadmap, also called the reduced visibility graph (RVG). The pursuer does not know which among the possible paths over the RVG the evader will choose, but the pursuer is free to move within all the environment. We provide a constructive method to solve the decision problem of determining whether or not the pursuer is able to maintain strong mutual visibility of the evader. This method is based on an algorithm that computes the safe areas (areas that keep evader surveillance) at all times. We prove decidability of this problem, and provide a complexity measure to this evader surveillance game; both contributions hold for any general polygonal environment that might or not contain holes. All our algorithms have been implemented and we show simulation results.

Keywords Pursuit-evasion · Tracking · Motion planning · Decidability · Complexity

A preliminary version of a portion of this work has been presented at the IEEE International Conference on Robotics and Automation (Murrieta-Cid et al. 2008).

✉ Israel Becerra
israelb@cimat.mx

Rafael Murrieta-Cid
murrieta@cimat.mx

Raul Monroy
raulm@itesm.mx

Seth Hutchinson
seth@illinois.edu

Jean-Paul Laumond
jpl@laas.fr

[1] Centro de Investigación en Matemáticas, CIMAT, Guanajuato, Mexico

[2] Tecnológico de Monterrey, Escuela de Ingeniería y Ciencias, Atizapan, Estado de México, Mexico

[3] University of Illinois at Urbana-Champaign, Urbana, IL, USA

[4] LAAS-CNRS, University of Toulouse, Toulouse, France

1 Introduction

The problem we consider in this paper is related to pursuit-evasion games. A great deal of previous research exists in this area of pursuit-evasion, particularly in that of dynamics and control in the free space (without obstacles) (Hájek 1965; Isaacs 1965; Başar and Olsder 1982). The pursuit-evasion problem is often framed as a problem in noncooperative dynamic game theory (Başar and Olsder 1982).

Pursuit-evasion can be defined in several ways. One formulation requires *finding* the evader with one or more mobile pursuers that sweep the environment so that the evader does not eventually sneak into an area that has already

been explored. Deterministic (Parsons 1976; Suzuki and Yamashita 1992; Guibas et al. 1999; Sachs et al. 2004; Tovar and LaValle 2008; Barrière et al. 2002) and probabilistic (Vidal 2002; Hespanha et al. 2000; Isler et al. 2005; Chung 2008; Hollinger et al. 2009) algorithms have been developed in this vein. A recent survey of this kind of problems in mobile robotics is reported in Chung et al. (2011).

Alternatively, the pursuer(s) might have as a goal to actually "catch" the evader(s), that is, move to a contact configuration, or closer than a given distance. An example of this kind of problems is the classical differential game, called the homicidal chauffeur (Isaacs 1965; Merz 1971). There, a faster pursuer (w.r.t. the evader) has as its objective to get closer than a given constant distance (the capture condition) from a slower but more agile evader.

These problems are different from ours, since we assume that the pursuer is initially aware of the evader's position, with the goal of maintaining sight of the evader. Indeed, in this paper, we consider the problem of making a mobile robot, called the *pursuer*, maintain strong mutual visibility (SMV) of a moving *evader*. SMV is a visibility notion between regions which requires a convex partition of the environment. The environment is filled with obstacles, the speed of each of the participants is bounded and the participants may be separated by an arbitrary distance.

We have already developed motion strategies for evader surveillance in Murrieta-Cid et al. (2007), analyzing two main scenarios: one where the distance between the pursuer and the evader is variable but the speed of both players is unbounded; and other where the speed of both the evader and the pursuer is bounded but the distance between the pursuer and the evader is constant. Under the latter assumption a slower pursuer will always be defeated by the evader, even in an environment without obstacles. In this paper, we present a more general formulation, in which, we simultaneously consider that the players move at bounded speed, but that there is no constraint upon their separating distance. Now a slower pursuer may be able to maintain surveillance of the evader in a polygonal environment which is simply connected. A careful inspection on the map and the initial position of both participants is required to determine the existence of a solution. In this paper we provide such analysis.

In Murrieta-Cid et al. (2007), we were able to establish sufficient conditions for escape by the evader (note that if the evader can escape an infinitely fast pursuer, then it will naturally escape a pursuer with finite speed), but we were unable to determine sufficient conditions under which the pursuer could maintain visibility of the evader. In the present paper, we provide sufficient conditions for surveillance by exploiting the concept of SMV between regions in a convex decomposition of the environment.

Furthermore, in our previous work, we did not consider the combinatorial problem inherent to any strategy that considers visiting several locations in an environment with obstacles, for the case of bounded pursuer and evader speeds. In this paper, we solve this problem.

We note that our approach relies on the choice of a specific convex decomposition of the environment (indeed, our notions of visibility rely on this decomposition). Different decompositions will lead to different solution strategies for the pursuer. Therefore, in Appendix 2 we propose specific algorithms to construct a decomposition that enjoys a number of favorable properties. In short, we provide two main contributions: (1) we prove decidability of this problem for any arbitrary polygonal environment, which, as far as we know, has not been done before for the target tracking problem, and (2) we provide a complexity measure to our evader surveillance game.

The remainder of this paper is organized as follows. Section 2 describes related work. Section 3 defines SMV that is a notion of visibility among convex regions in a given environment partition, and also introduces what we call the mutual visibility graph and the accessibility graph (AG), which respectively will help us to make queries of SMV and accessibility between regions in the given partition. Once strong mutual visibility is defined, Sect. 4 makes the formal statement of our problem. Section 5 provides some basic definitions and an overview of the proposed solution to the addressed surveillance problem. In particular the concept of *safe areas* is introduced, which are subsets of the workspace where the pursuer must lie to maintain SMV of the evader, taking into account the constraint propagation over all the possible paths that the evader can travel. Section 6 shows two algorithms to compute the safe areas for any given environment; the first one (Sect. 6.1) used when the environment RVG has a tree topology, and the second one (Sect. 6.2) used when the environment RVG has cycles within it. Section 7 depicts the surveillance motion strategy for both the evader and the pursuer. For the evader all possible paths are considered, and for the pursuer its motion strategy is based on the safe areas. Section 8 offers a deeper discussion of the presented problem regarding to its decidability along with its complexity, also providing a proof of convergence of the algorithm presented in Sect. 6.2, all of that, making use of a modeling of our problem based on infinite compositions of relations (Takeuti and Zaring 1971). In Sect. 9, we present simulation results, and in Sect. 10 the conclusions and future work. In the Appendices we show a proposal of a convex partition which enjoys some useful properties for the pursuit-evasion problem. Finally, to facilitate the presentation of this work, at the end of the manuscript we provide a table with the used notation along this work.

2 Related work

Recent years have seen a growing interest in the problem of maintaining visibility of a moving evader in an environment with obstacles (LaValle et al. 1997; González et al. 2002; Jung and Sukhatme 2002; Bandyopadhyay et al. 2006; Bhattacharya and Hutchinson 2009). Game theory is proposed in LaValle et al. (1997) as a framework to formulate the tracking problem, and an online algorithm is presented. In Becker et al. (1995), an algorithm is presented that operates by maximizing the probability of future visibility of the evader. This algorithm is also studied with more formalism in LaValle et al. (1997). The work in Fabiani and Latombe (1999) presents an approach that takes into account the positioning uncertainty of the robot pursuer. The approach presented in Murrieta-Cid et al. (2005) computes a motion strategy by maximizing the *shortest distance to escape*, this is the shortest distance the evader needs to move in order to escape the pursuer's visibility region. In González et al. (2002), a technique is proposed to track an evader without the need of a global map. Instead, a range sensor is used to construct a local map of the environment, and a combinatorial algorithm is then used to compute a motion for the pursuer at each iteration. Also in González et al. (2002), it has been noticed that particular difficult configurations arise when the line of sight between the evader and the pursuer is in contact with an obstacle. Improving upon González et al. (2002), Bandyopadhyay et al. (2006) presented an approach based on a local minimum risk function to deal with such configurations. In Efrat et al. (2003), the authors show how to efficiently (low-polynomial) compute an optimal reply path for the pursuer that counteracts *a given evader movement*, however, differently to the work presented in this paper, Efrat et al. (2003) does not deal with the problem of deciding whether or not there is an evader path that escapes surveillance. In O'Kane (2008), a robot has to track an unpredictable target with bounded speed; the robot's sensors are manipulated to record general information about the target's movements, but avoiding that detailed information about the target's position is available whenever the robot's sensors are accessed by other agent that can damage the target.

The work presented in Bhattacharya and Hutchinson (2009) addresses the problem of maintaining classical visibility of the evader as a *game of degree* (i.e., the emphasis is over optimizing a given criterion and not over the problem of deciding which is the winner player). The pursuer and the evader are omnidirectional (holonomic) systems in an environment containing obstacles. That work proves the existence of players strategies that are in Nash equilibrium; the pursuer wants to maintain visibility of the evader for the maximum possible time, while the evader wants to escape the pursuer's sight as soon as possible. The work in Bhattacharya and Hutchinson (2009) presents necessary and sufficient conditions for the visibility based target tracking game in conjunction with the equilibrium strategies for the players. However, in Bhattacharya and Hutchinson (2009) the authors do not consider long combinatoric evader paths that take into account the interaction of several obstacles (see Fig. 4), as the authors consider one single obstacle corner at once.

Others have studied an extended version of the problem involving multiple participants of each kind (evaders and pursuers). For example, Parker (2002) developed a method that attempts to minimize the total time in which the evaders escape surveillance. In a similar vein, Jung and Sukhatme (2002) combined the application of mobile and static sensors, using a measure of the degree of occlusion of the evaders. In Barrière et al. (2002) the authors consider a team of mobile agents deployed to capture an intruder in a network. In that work the authors proposed an algorithm that determines the minimum number of agents to capture the intruder in a tree network. The authors use a message-passing architecture as in this work, however, the meaning of the message is different since the authors do not consider a polygonal environment.

Almost all existing work focuses on the 2-D version of the problem of maintaining visibility of an evader, but there are just some few works that deal with the 3-D version of it, mainly because of the complexity of the visibility relationships in 3-D. One work that deals with the 3-D version of this problem is the one presented in Bandyopadhyay et al. (2007). Here the authors present an online algorithm for 3-D target tracking among obstacles, using only local geometric information available to a robot's visual sensors. To prevent the target from escaping from the robot's visibility region both in a short and long terms, a risk function is efficiently computed. The robot motions are calculated minimizing the risk function locally also in a greedy fashion. By one hand, the greedy approaches such as González et al. (2002), Bandyopadhyay et al. (2006), Bandyopadhyay et al. (2007) have the advantage of providing practical solutions with low computational cost. But on the other hand, they have the disadvantage that they may loose solutions since they do not consider long term paths travelled by the evader, as in our approach. Furthermore, our proposed algorithms can be adapted to handle short term planning horizons (see Sect. 9), even to the point of obtaining a totally one step ahead greedy approach, but with the consequence of loosing solutions.

Maintaining visibility of a moving agent may be used in a variety of applications. For example, in Tekdas et al. (2010), 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. That work also includes a proof-of-concept implementation. Similarly, in Stump et al. (2011) the authors consider the problem of deploying robots in formations that ensure network connectivity between a fixed

base station and a set of independent agents wandering in the environment. The authors used a communication model that requires line-of-sight. They solved robots placements by finding mutually-visible configurations in a polygonal decomposition of the environment map.

More related to the presented work is the one in Bhattacharya and Hutchinson (2011). In that work the problem of maintaining visibility of a moving evader is addressed as a *game of kind* (deciding which player wins). The authors provide guaranteed strategies for surveillance for the observer in an environment containing a single corner. Later, they extend their results for a second case of a general environment containing polygonal obstacles. The evader travels the shortest paths [equivalent to traveling over the reduced visibility graph (RVG)] to convex corners, while the pursuer aims to be placed on regions of the workspace (i.e., star region) related to the respective convex corners in order to prevent the evader from escaping. In the second case the authors provide a set of starting points for the pursuer where the winner of the game is known. However, there is another set which the authors point out that the winner of the game is unknown, hence, in that set the decision problem is unsolved. Similarly to the approach presented in Bhattacharya and Hutchinson (2011), in this work the evader also travels the RVG, and also specific goals for the pursuer are established for preventing the evader from escaping. As a step beyond the results presented in Bhattacharya and Hutchinson (2011), in the present work we provide complete decidability of the problem for any arbitrary polygonal environment. Furthermore, in Bhattacharya and Hutchinson (2011) the authors do not consider that the surveillance constraints propagation (given by the evader's decision of visiting corners with a given order in an attempt to escape) increases the difficulty for the pursuer to maintain surveillance. They neither consider long term combinatorial paths that might include cycles. In this work we show that it is not possible to decide which player wins considering corners independently, for this reason, we consider both the surveillance constraints propagation and long term combinatorial paths. Besides, we show that the problem is decidable even if the evader travels in a cycle forever.

In Murrieta-Cid et al. (2008) the notion of SMV was introduced. SMV is a notion of visibility between regions. In Murrieta-Cid et al. (2008) it was shown that when we know the evader trajectory in advance, assuming given pursuer and evader positions, if the evader moves over the shortest path to a region R that is not visible for the pursuer, and at all time the pursuer is able to reach a region that is strongly mutually visible with the region where the evader lies, then the evader is not able to escape making use of the shortest path to reach R or any other path that is not the shortest one. In this paper we take a step further, namely, we assume an antagonistic evader who moves continuously and unpredictably, but with a constraint over its set of admissible motion policies, as the evader moves in the shortest-path roadmap, the RVG. Notice that *the pursuer does not know, which among the possible paths in the RVG, the evader will choose*; as a consequence, the main difficulty to address in a solution for that scenario is the need to consider *all* such possible evader's paths at the same time.

In spite of these efforts, the decision problem answering whether or not the evader can escape, has not been addressed for the case in which the evader moves traveling long combinatoric paths in an environment with obstacles, while the speed of each participant is bounded and the surveillance distance varies. Answering this question under a SMV framework is one of the goals of this paper.

3 Strong mutual visibility

We now introduce the definition of SMV. First, let $R_1, \ldots, R_n$ be a convex partition of the environment (Latombe 1991), i.e., each R_i is a convex polygon, the workspace $W = \bigcup_i R_i$, and $int(R_i) \bigcap int(R_j) = \emptyset$ for $i \neq j$.

Once we have a convex partition of the environment, we establish a type of visibility between regions, which we call strong mutual visibility. SMV is a binary relation that holds when all points in two regions are mutually visible to one another.

Definition 1 Two regions R and R' are said to be *strongly mutually visible* if classical visibility[1] holds for all points x and x' such that $x \in R$ and $x' \in R'$.

SMV, defines a stronger condition than visibility between pairs of points (classical visibility (O'Rourke 1987)). A straightforward test for SMV using a convex hull computation (O'Rourke 2000), is given in the following expression:

Regions R and R' are strongly mutually visible if and only if

$$\text{int}\big[\text{convex} - \text{hull}\big[(R \cup R')\big]\big] \subset W$$

where W is the polygon representing the workspace.

A given partition of the environment into convex regions induces an *AG* and a *Mutual visibility graph* (*MVG*). In each graph, nodes represent regions. In the 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 the 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. Figure 1b and c respectively show the MVG and the AG associated to the partition of

[1] In classic visibility two points are visible when a line segment between them does not intersect any obstacle Shermer (1992).

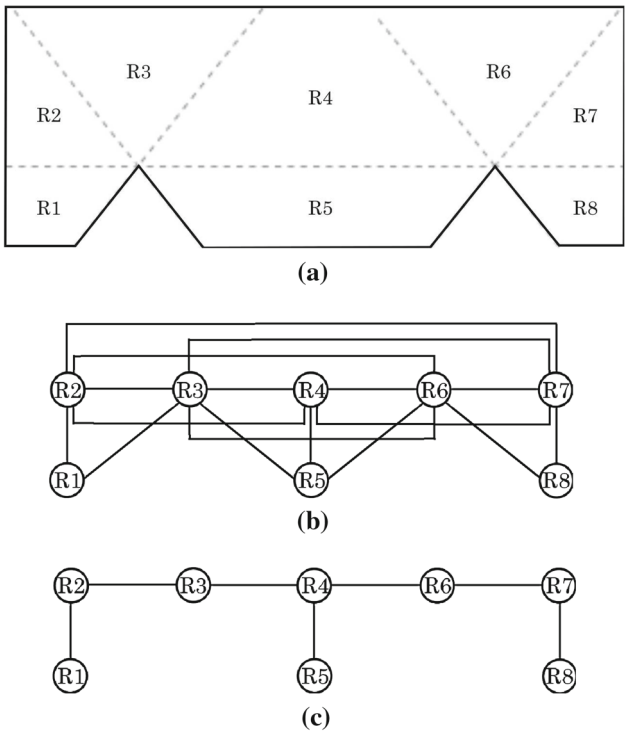


Fig. 1 **a** Environment partition, **b** mutual visibility graph, and **c** accessibility graph

the environment shown in Fig. 1a. The MVG therefore provides information to find a sufficient condition to maintain evader visibility while an AG provides adjacency between regions.

In the remaining of this work we use the convex partition presented in Appendix 2 to explain and illustrate our method. Nevertheless, we stress that our method works with any other convex partition. However, the partition presented in Appendix 2 has the next convenient properties.

One, the decomposition includes regions (corner-guard polygons) that allow SMV to be maintained at all regions adjacent to reflex vertices[2] (reflex vertices being the main source of difficulty for maintaining visibility). This gives a place to stand for the pursuer in order to prevent the evader to escape using individually each reflex vertex. Two, our decomposition maximizes a measure of SMV for the adjacent regions (see Theorem 5 in "How to refine a given convex partition (pivot segments)" section in Appendix 2). Intuitively, this property means that for a given number of regions, the total area visible to the players is maximized. If any other partition is used to obtain the same visible area, more regions would be needed, which implies more computation. Three, our decomposition approach admits a refinement mechanism, which allows the construction of decompositions at arbitrary resolution, such that SMV tends to classical visibility as the number of regions increases [see "How to refine a given convex partition (pivot segments)" section in Appendix 2].

[2] A reflex vertex is a polygon vertex of an internal angle greater than π.

4 Problem statement

The evader and the pursuer are modeled as points moving over a known environment. The environment is modeled as a polygon that might or not contain holes. Every participant is assumed to accurately know its position at all times, and is limited to move at bounded speed (we denote V_e as the evader's maximum speed, and equivalently, V_p for the pursuer). Other than these, no kinematic nor dynamic constraints are imposed on the pursuer or the evader.

The pursuer moves freely within the workspace. We shall assume that the evader will move over the RVG. The RVG of a polygonal environment, also known as the shortest-path roadmap, is built in the next way: first of all, the vertices in the RVG are the reflex vertices of the polygonal environment. Second, the edge between two vertices in the RVG is generated if the two vertices are endpoints of the same edge of an obstacle, or if a bitangent line can be drawn between such vertices (LaValle 2006). As the reflex vertices break the convexity of the environment, then a reflex vertex is associated to an escape path. The RVG is of interest as it includes the shortest paths to any reflex vertex in the environment. Furthermore, for an evader that is free to move in the workspace, the RVG still gives the shortest path to reach any place within the polygonal environment, therefore, in most cases, it is the roadmap that is most advantageous to the evader. Thus, the RVG represents a worst-case scenario for the pursuer.

Both players are equipped with an omni-directional sensor. We use a SMV model, where the pursuer maintains SMV with the evader if at all times the pursuer and evader lie in regions that are SMV. Clearly, maintaining SMV is a *sufficient* condition for classical visibility, however, it is not a necessary one, as can be easily seen in Fig. 2.

In classical visibility, pursuer-evader configurations where the line of sight between the evader and the pursuer is in contact with an obstacle, are difficult to study analytically, and can be a chief impediment in the search for necessary and sufficient conditions, to decide whether or not the pursuer can maintain visibility of the evader. In this particular case, it is not clear what the pursuer should do; there is a conflict as to what the pursuer should strive towards: either minimizing the shadow region so as to prevent escaping or minimizing the distance so as to prevent a further, second occlusion [see Fig. 2, this issue has been already noticed in González et al. (2002)]. As will become evident, this conflict does not arise for the case of maintaining SMV, bringing interest on using SMV. Reasoning in terms of maintaining visibility of regions

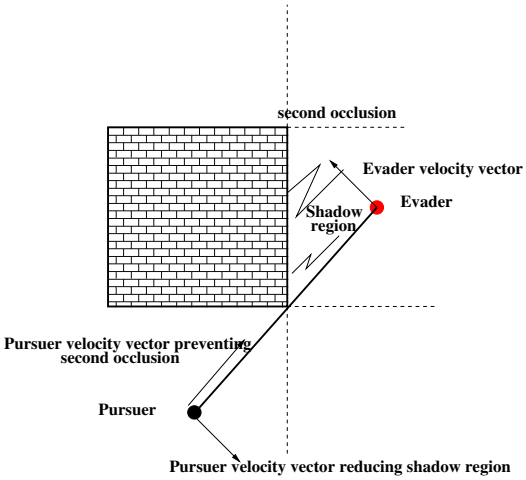


Fig. 2 Classical visibility

represents to plan to prevent the evader from taking the players to configurations as shown in Fig. 2, which are very hard for the pursuer. Furthermore, as will be shown in Appendix 2, SMV is a solution for classical visibility for a given resolution, since as we have already mentioned, SMV tends to classical visibility as the number of regions increases in an appropriate partition. Such partition is also provided (see Appendix 2).

Considering the past scenario we address our target tracking problem as a game of kind (Başar and Olsder 1982) consisting in the next decision problem: *is the pursuer able to maintain surveillance—in the sense of SMV—of an evader at all time*?

Any instance of our decision problem is given by the tuple $(V, A, \{R_i\}, V_{ratio}, \Lambda, \Pi)$, where V and A represent respectively the nodes and the edges of the reduced visibility graph, that is $RVG = (V, A)$, $\{R_i\}$ is the partition of the environment, $V_{ratio} = V_p/V_e$ is the speed ratio of the players, Λ is the size of the paths that the evader travels in terms of the number of sub-goals (e.g. reflex vertices) that it visits, and Π the set of particular properties that define the evader's paths over the RVG (for instance, properties that make the paths Hamiltonian as in Sect. 8.2).

In the next sections we build over the most general instances of our problem, where the RVG may have cycles or not, for any convex partition $\{R_i\}$ and any V_{ratio} (we might have players with equal velocities or either player might be faster), evader paths of any size (Λ) and with no particular properties (Π) over them apart from being over the RVG.

5 Basic definitions and overview of solution

Our problem is a non-cooperative game because the evader and pursuer have antagonistic goals (Rappoport 1966); the evader wants to escape pursuer's SMV and the pursuer wants to maintain evader surveillance. By convention, the visibility constraint is initially satisfied. Therefore, an action must be taken by the evader to break it. If the evader is in a region SMV with the region where the pursuer lies, then the evader must travel to a region that is not SMV with the region where the pursuer is located. To determine the regions not SMV to the pursuer's region the MVG is used. Similarly, to establish the connectivity between regions the AG is queried.

Let E be the region where the evader is located, and P the region where the pursuer lies. For each region R_i in the environment partition there is a set of regions that is SMV to R_i, let us call it $SMV(R_i)$. To maintain surveillance, the pursuer must be in a region belonging to $SMV(E)$ at every instant of time, therefore, let us assume that $P \in SMV(E)$. If the evader transits from a region E to a region E', then in order to maintain evader's surveillance the pursuer must move to a P' such that $P' \in SVM(E')$. While the evader travels over the RVG there are some critical locations that correspond to frontiers between regions in which the set $SMV(E)$ changes. Such punctual critical locations are what we call critical points. More formally a critical point is defined as follows.

Definition 2 A *critical point* is a point $q \in$ RVG such that when the evader reaches (or leaves) it, the evader transitions between regions, hence, $SMV(E)$ changes.

There are two types of critical points. The first type of critical points are the *reflex vertices*. The other type are the *transition points*, which are critical points located on the RVG edges and are not reflex vertices.

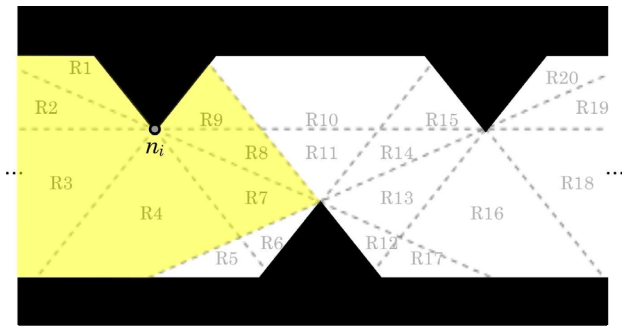
A critical point belongs to more than one region at a time. When the evader is at a critical point q, the pursuer must lie in a region that is SMV to all the regions to which the critical point belongs, that is, P must be in the set of all regions which are all mutually visible to all the regions that own the point q, which we denote as the guard polygon $gP(q)$. More formally, the definition of a guard polygon is as follows:

Definition 3 A *guard polygon* for a given point q is the set of all regions in which each of them is mutually visible to all the regions that own the point q. Let $Q(q) = \{R : q \in R\}$, then a guard polygon $gP(q)$ for a given point q is defined by:

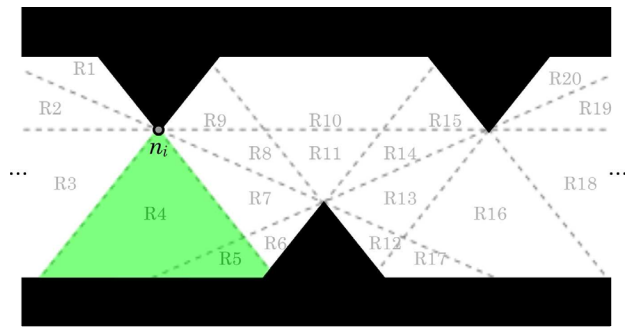
$$gP(q) = \left\{R : (R, R_k) \in MVG, \quad \forall R_k \in Q(q)\right\} \tag{1}$$

Figure 3b depicts an example of a guard polygon for a critical point of a reflex vertex type.

Furthermore, evader surveillance must be guaranteed at all times and not only at critical points, therefore, surveillance must be maintained while the evader is traveling between critical points, namely, over edges of the RVG. The edges of



(a) If the evader stands on n_i, it is simultaneously over regions $\{R1, R2, R3, R4, R7, R8, R9\}$



(b) The guard polygon for point n_i is $gP(n_i) = \{R4, R5\}$

Fig. 3 Guard polygon

the RVG might also belong to more than one region of the partition. In Sect. 6 we present an algorithm that guarantees evader surveillance at all time, given that the evader is on the RVG, either on critical points or traveling between them.

Remark 1 As we have already mentioned, the evader might be located in a point that belongs to more than a single region. This indetermination of the evader location can be considered as uncertainty related to the region in which the evader is. When assumed that the evader is in a given location, in order to maintain SMV of it, the pursuer must maintain SMV of all the regions that contain the evader location.

The evader has critical points as goals to escape, while the pursuer has guard polygons as goals. There is a one to one relation between each critical point to a respective guard polygon. To maintain evader surveillance as it visits a set of critical points, the time that takes to the evader to reach them must be strictly greater than the time that takes to the pursuer to reach the respective set of guard polygons. Assume that the evader visits a sequence of critical points $q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow \cdots$, let t_e^k be the smallest time that takes to the evader to reach the critical point $k+1$ from its k critical point, and t_p^k the smallest time that takes the pursuer to reach its corresponding $k+1$ guard polygon departing from the k guard polygon. If $t_p^k - t_e^k \leq 0 \forall k$, then the pursuer is able to maintain surveillance, otherwise the evader escapes.

Unfortunately, the sequence of time restrictions imposed by the evader cannot be analyzed by independent pairs as will be shown in Sect. 8.1, since the set of points where the pursuer must be at time k to fulfill the surveillance restriction at that instant of time, will modify the available set of points at time $k+1$ where the pursuer can keep further evader's SMV. Such influence will keep propagating modifying the set of points where the pursuer is able to maintain evader surveillance during the whole path that the evader is free to choose. To model this constraint propagation over the points where the pursuer must be, we propose a *message propagation algorithm* in which the messages are the set of points that satisfies the constraints for the pursuer. The result of the message propagation algorithm are what we denote as *safe areas*, which are subsets of points of the guard polygons, where the pursuer must lie to keep evader surveillance taking into account the constraint propagation over all the possible paths that the evader can travel.

Additionally, to characterize the paths that the evader can travel, we use the reflex vertices as reference points. Indeed, the safe areas are related to the reflex vertices in the RVG, thus, each reflex vertex n_i has a related safe area denoted by $sA(n_i)$.

Remark 2 In general the evader travels semi-free paths, that is, it is allowed to move in contact with the obstacles.

The safe areas depend on the environment structure. To give a final answer to our decision question, the initial positions of the players are included to obtain a region of the workspace, which we call S set, where the pursuer must be located to maintain SMV at any instant of time given the current evader's position (see Sect. 7). The S set defines a motion strategy for the pursuer consisting on maintaining itself within that set.

It is worthwhile to mention that the proposed message passing algorithm is based in critical points over the roadmap, defined as points over graph edges that represent the risk of losing evader visibility (this is not unique to the RVG), and minimal time to travel between such critical points. Hence, our algorithm is applicable to other roadmaps provided that the minimal time to travel between critical points is computable.

In the remaining of this paper, we present an approach that is able to achieve the following specific results:

(1) Algorithms are presented such that for any pair of bounded maximal speeds of the players, and any pair of initial positions, these algorithms answer which player wins: the pursuer wins if it can maintain evader's surveillance under SMV at all time otherwise the evader wins (Sects. 6, 7).
(2) Based on the map only, and regardless of the initial position of the players, we can determine when the pursuer

will not be able to keep strong mutual visibility of the evader due to the environment structure (Sect. 7.1).
(3) If there is a solution for the pursuer, our approach can determine the part of the space—S set—where the pursuer must be at every instant of time to guarantee evaders surveillance (Sect. 7.2).
(4) We show that the decision problem posted above is decidable (Sect. 8.1).
(5) We provide a complexity characterization of this problem (Sect. 8.2).

In the next section we present the message propagation algorithm to calculate the safe areas.

6 Algorithms for computing safe areas

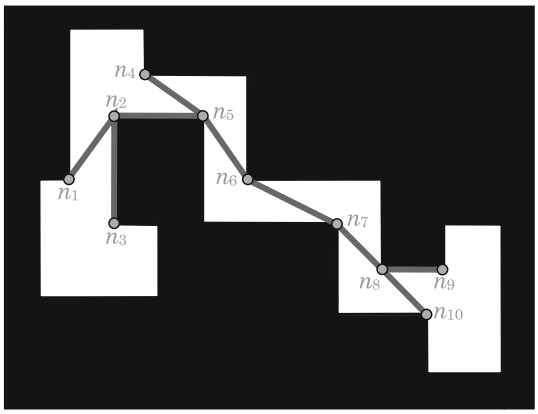
The RVG can be classified into two types: with or without cycles. To facilitate the exposition, we first present a procedure to compute the safe areas for the case when the RVG has a tree topology. Later, based on such procedure we present an algorithm to compute the safe areas when the RVG has cycles within its structure.

6.1 Tree shape RVG

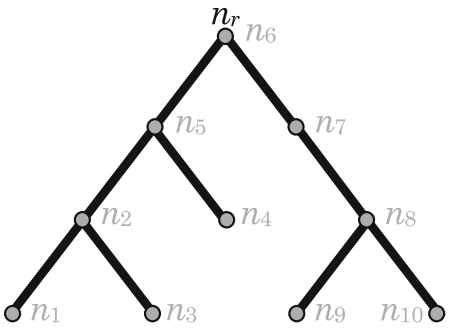
In this subsection we present an algorithm to compute the safe areas when we have a tree topology RVG. The first step of the presented algorithm is to root the RVG, making any $n_i \in RVG$ the root node n_r of the tree. The second step is to initialize each safe area $sA(n_i)$ with the guard polygon $gP(n_i)$ related to its respective reflex vertex n_i. In a third step, we proceed to refine the current value of the safe areas to consider the constraint propagation given by the long term paths that the evader can travel. The constraint propagation is done through a message propagation scheme, where starting from the leaf nodes on the rooted RVG, the children nodes will pass a message (a set of points that satisfies surveillance constraints, see below) to their parents in a bottom-up hierarchical fashion, repeating the procedure until n_r is reached; then the same procedure is performed in the opposite direction, starting from n_r all the way down until the leaf nodes are reached. The message passing scheme considers all the possible paths that the evader can travel. This is proved in Theorem 1 for any RVG, either with a tree topology or with cycles within it.

Theorem 1 *The proposed message-passing scheme considers all the possible paths of unrepeated nodes that the evader might travel in the RVG. Furthermore, it also considers any path of repeated nodes.:* Refer to Appendix *1* for a proof.

Figure 4 shows a map with its related RVG over it, and the RVG rooted in n_6.



(a) Map and RVG



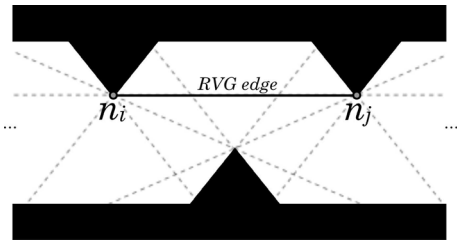
(b) Tree rooted in n_6

Fig. 4 Map with RVG tree topology

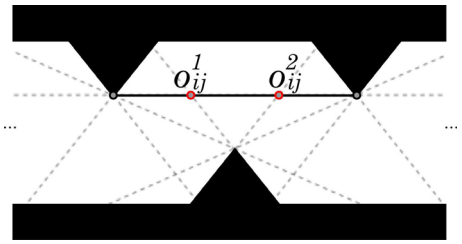
Now we are more precise of what are the propagated messages, but we first present some notation.

We divide each edge of the RVG into sub-edges, being the transition points the division points (see Fig. 5a and b). For convention, if there are no transition points within a RVG edge, then such edge has only one sub-edge. Let us consider the RVG edge between nodes n_i and n_j. In the next notation we consider an order from n_i to n_j. o_{ij}^k denotes the k-th transition point between n_i and n_j (see Fig. 5b). l_{ij}^k denotes the k-th sub-edge between n_i and n_j (see Fig. 5c). As the evader travels over an edge it visits a series of regions that can be grouped in clusters. Each cluster is generated grouping the regions that have a *common sub-edge*; then, let C_{ij}^k denote the k-th cluster between n_i and n_j (see Fig. 5d). Note that, as the evader travels in a given sub-edge the set of regions that are SMV with the evader is constant. Furthermore, C_{ij}^k is used for the pursuer to maintain SMV of the evader, when the evader traverses a sub-edge (see below). This notation with three indices (i, j, k) uniquely identifies each of the past elements over the RVG.

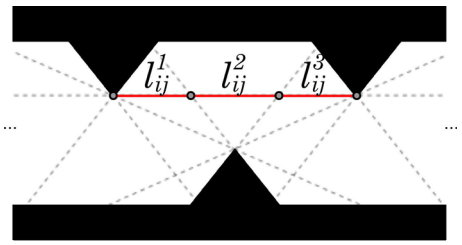
Based on the observations that while the evader travels in a given sub-edge the set of regions that are SMV with the evader is constant, and that there is a one to one relation between each sub-edge l_{ij}^k and each cluster of regions C_{ij}^k, it is



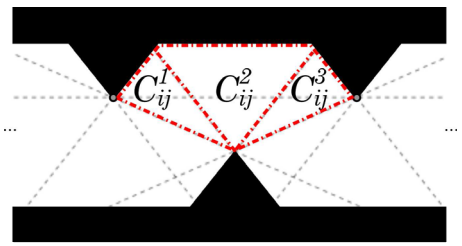
(a) Edge between n_i and n_j



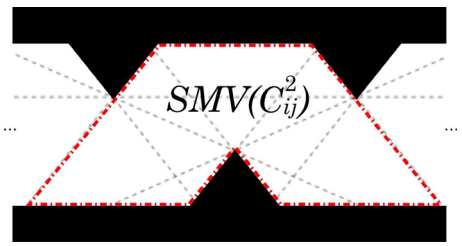
(b) Transition points



(c) Sub-edges



(d) Clusters of regions



(e) Set of regions SMV with cluster C_{ij}^2

Fig. 5 Features related to RVG edge between node n_i and n_j

possible to establish the following definition. $SMV(C_{ij}^k) = \{R : (R, R') \in MVG, \ such \ that \ \ l_{ij}^k \in R'\}$. This is, the set of regions SMV with the regions that fully contain a given sub-edge, see Fig. 5e. The set $SMV(C_{ij}^k)$ is of interest because the pursuer must be located into $SMV(C_{ij}^k)$ to maintain SMV of the evader, when the evader travels over l_{ij}^k.

We now describe the basic operation for sending a message from a node n_j to a node n_i. Let assume that between n_i and n_j there are no transition points, and that n_j is a leaf node in the rooted RVG. The basic operation consists in generating a reachable area over $SMV(C_{ij}^1)$ around $gP(n_j)$, and subsequently intersect the reachable area with $gP(n_i)$. The reachable area depicts the set of departing points belonging to $SMV(C_{ij}^1)$, from which the pursuer can reach a point in $gP(n_j)$ while the evader travels from n_i to n_j.[3] It is important to consider as points to be reached, only points over $SMV(C_{ij}^1)$, because this guarantees that the pursuer maintains SMV of the evader when the evader moves over the sub-edge l_{ij}^1. The set of points that fulfills the described constraints is called a *message*. The set of points (the message) sent from node n_j, is then intersected with the guard polygon $gP(n_i)$. The result of this intersection is the current safe area $sA(n_i)$ of node n_i. If there are transition points over the edge joining n_i and n_j, the same procedure is done propagating the message through all the transition points until n_i is reached, generating the respective reachable areas and performing the respective intersections over the guard polygons related to the transition points. If there are more leaf nodes connected to n_i apart from n_j, n_i must incorporate all the constraints transmitted by all the leaf nodes. For each leaf node a safe area for node n_i is generated and then the intersection of them is the current safe area $sA(n_i)$.

As mentioned above, the messages will first flow from the leaf nodes all the way up to the root node n_r, and then, from the root node all the way down to the leaf nodes, completing then the correct calculation of all the safe areas for all nodes in the RVG. The complete procedure is described by Algorithm 1 aided by Subroutines 2–5, being Algorithm 1 the main entrance of the procedure. In Algorithm 1, step 2. corresponds to the flow of messages from the leaf nodes all the way up to the root node, and step 3. corresponds to the flow of messages from the root node all the way down to the leaf nodes. Subroutines 2 and 3 are implemented as recursive depth first searches (Cormen et al. 2001), with the difference that Subroutine 2 does a post-order (Cormen et al. 2001) node processing while Subroutine 3 performs a pre-order (Cormen et al. 2001) processing of the nodes. Subroutines 4 and 5 give further details on how to implement the function that sends a message from a node n_j to a node n_i.

6.2 RVG with cycles

In this subsection we address the case of RVGs that have cycles intrinsic to their topology (see Fig. 6).

[3] In the presented algorithms the computation of the reachable areas is done backwards from n_j to n_i.

Algorithm 1 Computing Safe Areas sA

Input: Work space W, environment partition, RVG.
Output: Safe Areas sA.
for every node n_i in the RVG **do**
1. Initialize $sA(n_i) \leftarrow gP(n_i)$;
end for
2. $sA \leftarrow ReceiveChildNodesMessages(n_r, sA)$
3. $sA \leftarrow SendMessageToChildNodes(n_r, sA)$
4. Store(sA, RVG);

Subroutine 2 $ReceiveChildNodesMessages(n_i, sA)$

Input: $n_i \in RVG$ is the node that receives the message from its children, sA is the current set of safe areas.
Output: The modified set of safe areas sA.
if n_i is a $leafNode$ **then**
Return sA;
else
for every $n_j \in children(n_i)$ **do**
1. $sA \leftarrow ReceiveChildNodesMessages(n_j, sA)$;
2. $sA(n_i) \leftarrow sA(n_i) \cap GetMessageFromTo(n_j, n_i, sA)$;
end for
end if
Return sA;

Subroutine 3 $SendMessageToChildNodes(n_i, sA)$

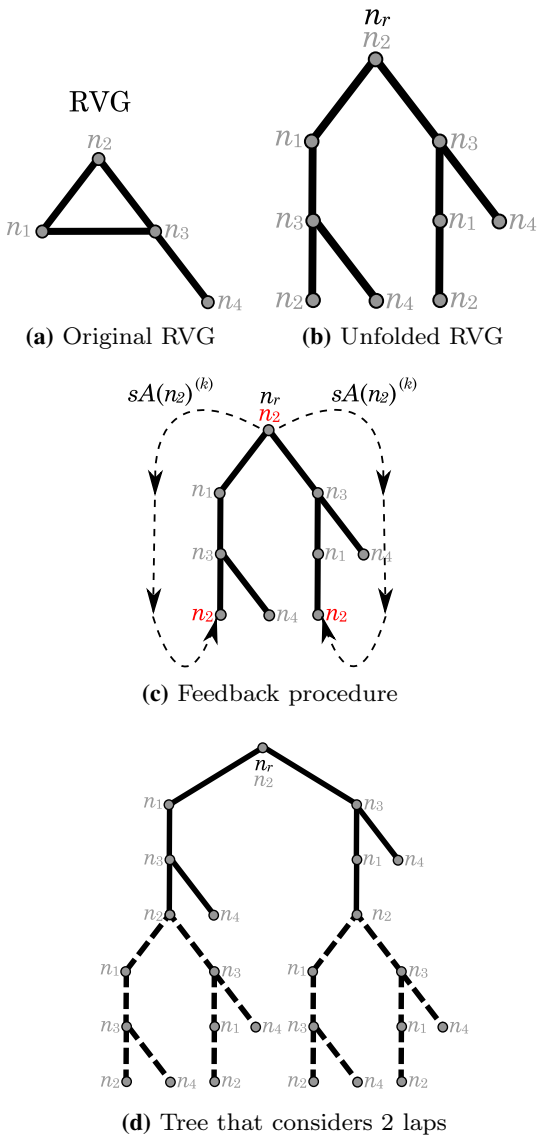
Input: $n_i \in RVG$ is the node that sends the message to its children, sA is the current set of safe areas.
Output: The modified set of safe areas sA.
if n_i is a $leafNode$ **then**
Return sA;
else
for every $n_j \in children(n_i)$ **do**
1. $sA(n_j) \leftarrow sA(n_j) \cap GetMessageFromTo(n_i, n_j, sA)$;
2. $sA \leftarrow SendMessageToChildNodes(n_j, sA)$;
end for
end if
Return sA;

Subroutine 4 $GetMessageFromTo(n_j, n_i, sA)$, gets the message from n_j to n_i

Input: Connected nodes n_j and n_i in the RVG, sA is the current set of safe areas.
Output: Message to intersect with $gP(n_i)$.
1. $o_{ij} \leftarrow GetTransitionPoints(n_i, n_j)$ –see Note at the end–;
if o_{ij} is not empty **then**
2. $Q \leftarrow gP(o_{ij}^N) \cap GetReachableArea(o_{ij}^N, n_j, sA(n_j))$;
if $N \geq 2$ **then**
for $k = N$ to 2 **do**
3. $Q \leftarrow gP(o_{ij}^{k-1}) \cap GetReachableArea(o_{ij}^{k-1}, o_{ij}^k, Q)$;
end for
end if
4. $A \leftarrow GetReachableArea(n_i, o_{ij}^1, Q)$;
else
5. $A \leftarrow GetReachableArea(n_i, n_j, sA(n_j))$;
end if
Return A;
Note: o_{ij} is the ordered set of N transition points between n_i and n_j.

Subroutine 5 $GetReachableArea(a, b, rB)$, generates a reachable area around rB

Input: Point a and point b (either a or b are transition-points or reflex vertices), $rB \subseteq gP(b)$.
Output: Reachable area.
1. $l_{ij}^k \leftarrow GetSubEdgeConnecting(a, b)$;
2. $C_{ij}^k \leftarrow GetClusterContainingSubEdge(l_{ij}^k)$;
3. $M = \{q \in SMV(C_{ij}^k) : d(q, x) \leq length(l_{ij}^k)\frac{V_p}{V_e}$ for some $x \in rB$, where $d(q, x)$ is the shortest distance within $SMV(C_{ij}^k)$ $\}$;
Return M;



(a) Original RVG **(b)** Unfolded RVG

(c) Feedback procedure

(d) Tree that considers 2 laps

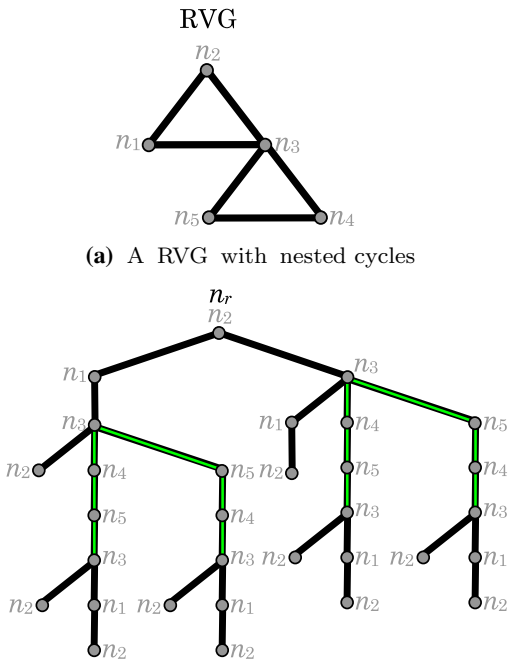
Fig. 6 Cycles algorithm

Cycles can be unfolded into sequences, taking advantage of that property, we can use the method presented in the last section to calculate the safe areas. For computing each safe area $sA(n_i)$, we repeat the next procedure for each node n_i in the RVG until convergence. (Section 8.1 presents formal arguments showing that the convergence is reached in a finite number of iterations.)

In a first step we unfold the RVG into a tree. Such transformation is made using a spanning tree (Cormen et al. 2001), performing a depth-first search from n_i, making it the root node n_r of the tree. From any node n_j already in the tree, the edges in the original RVG considered to continue the expansion, are the ones that fulfill the next properties: one, that they are different from the edge that lead us to n_j, and two, that they do not cause us to enter into a second lap in a cycle in the RVG or to enter in a cycle that can be considered in parallel by other branch in the tree. If there is no edge connected to n_j that fulfills both properties, then n_j is a leaf node in the tree. An example of such transformation is given in Fig. 6a and b. The resulting tree depicts all the possible paths that the evader can follow giving at most one lap into cycles. Then, over the resulting tree we apply the message passing scheme only from the leaf nodes all the way up to n_r.

To deal with the fact that the evader can travel several laps around a cycle, when the message passing scheme reaches the root node n_r, a feedback procedure takes place in every leaf node n_j equal to n_r (that is where a new lap over the cycle starts again), namely, we make the safe area of each n_j equal to the current value of the safe area of n_r; then the message propagation flows again toward n_r. An example of such procedure is presented in Fig. 6c. In Fig. 6b it can be seen that a cycle begins in n_r and the cycle is closed in two of the leaf nodes (n_r and the two mentioned leaf nodes refer to n_2 in the original RVG in Fig. 6a). At iteration k of the feedback procedure, $sA(n_r)^k$ is generated, which is the safe area related to node n_r considering that the evader travels at most k laps over cycles. Figure 6d shows an equivalence between applying once the message-passing scheme in the tree shown in that figure and applying the feedback procedure for $k = 2$ in the tree shown in Fig. 6c. Finally, we keep iterating through this feedback process until we observe no further change between $sA(n_r)^k$ and $sA(n_r)^{k+1}$. This stop condition is always achieved as shown in Sect. 8.1.

Depending on the RVG structure, there might be nested cycles, as shown in Fig. 7. In such cases the convergence is first sought in the most inner cycles, and hierarchically climbs up to the most outer cycle. In the example shown in Fig. 7b, the outer cycle that starts in n_2 and returns to n_2 has a nested inner cycle that starts in n_3 and returns to n_3. For each iteration k of the feedback procedure over the outer cycle, several iterations over the inner cycle are performed until the stop condition is fulfilled, and the procedure keeps iterating until the stop condition is also fulfilled in the outer cycle.



(a) A RVG with nested cycles

(b) Tree representation unfolding the RVG with nested cycles

Fig. 7 Nested cycles

Algorithm 6 Computing Safe Areas for Cycles sA

Input: Work space W, environment partition, RVG.
Output: Safe Areas sA.
for every node n_i in the RVG **do**
1. Initialize $sA(n_i) \leftarrow gP(n_i)$;
2. $Iterations(n_i) \leftarrow 0$;
end for
3. $sA \leftarrow ReceiveChildNodesMessagesC(n_r, sA)$
4. Store(sA, RVG);

Algorithm 6 gives the details about the presented feedback procedure.

7 Pursuit/evasion strategies

7.1 Sufficient condition for escape based on the map

In the past sections we have presented a complete procedure for calculating the safe areas $sA(n_i)$ related to any node n_i within the RVG. Figure 8 shows a given environment with the two safe areas related to the two reflex vertices in the environment.

The safe areas might be calculated in a preprocessing stage before the actual pursuit-evasion game starts. Actually, just using the mere safe areas a winning criterion for the evader can be defined. If any calculated safe area results into an empty set, it is a sufficient condition for the evader to win

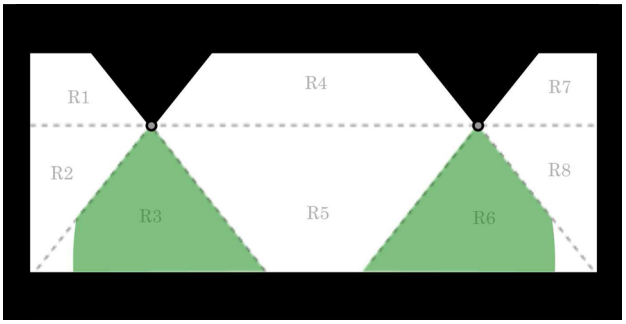


Fig. 8 Resulting safe areas

that depends just on the environment structure and not in the initial configurations of the players, so such condition can be verified before the game even starts. If the past condition is fulfilled it is not even worthwhile for the pursuer to play the game because it will inevitably lose, however, if the contrary thing happens, the game would take place and in such case we are interested on giving a motion strategy for the pursuer so that it can maintain mutual visibility of the evader. It is worthwhile to mention that the safe areas by themselves actually contain a whole family of possible solution paths for the pursuer, so using them we will proceed to establish a motion strategy that tells the pursuer where to be at any instant of time.

Remark 3 When any calculated safe area results into an empty set, it is a sufficient condition for the evader to win the game. This condition only depends on the environment structure so it can be verified regardless of the initial positions of the players.

7.2 Sets of local solution

If none of the safe areas results into an empty set, then the pursuer needs a motion strategy to keep surveillance of the evader. The presented pursuer's motion strategy makes use of what we call the S set (it comes from solution set), which will be a set of points within the environment, where the pursuer must be at any instant of time such that it is able to keep surveillance of the evader. The S set changes dynamically as the players keep moving. The pursuer's motion strategy consists on maintaining itself inside the current S set.

For the evader's strategy we proceed systematically, our algorithms consider *all* the possible paths that the evader can follow (see Theorem 1).

The S set is primarily a function of the position of the evader, denoted by e. If the evader is over a reflex vertex $n_i \in$ RVG, we already know that the pursuer must be located within $sA(n_i)$ to prevent an escape through n_i, and to enable the pursuer to reach any other safe area at the time that the evader arrives to the respective reflex vertex, satisfying any intermediate surveillance restrictions between nodes. In other words, when $e = n_i$, then $S = sA(n_i)$. A more complicated case is when e lies over an edge in the RVG. Let us suppose that e is in the edge between reflex vertices n_i and n_j, in that case the S set is given by (2).

$$S = \left\{ q : \; q \in gP(e) \bigwedge_{v \in \mathbf{V} = \{n_i, n_j\}} t_p(q, sA(v)) \leq t_e(e, v) \right\} \tag{2}$$

In (2), $\mathbf{V}$ is a set that contains the reflex vertices or nodes n_i and n_j. $t_e(e, v)$ refers to the smallest time that will take to the evader to travel from its current position e to point v. $t_p(q, sA(v))$ refers to the smallest time that will take to the pursuer to move from point q to reach the safe area $sA(v)$, taking into account that the pursuer must satisfy intermediate surveillance restrictions due to the fact that the evader may visit a series of transition-points while it moves towards v. Indeed, the S set is a safe area related to e, which would be calculated by generating reachable areas (messages) around $sA(n_i)$ and $sA(n_j)$ towards e, propagating the messages through a series of respective transition-points o_{ij}^k until e is reached; it could be interpreted like e is a virtual node within the RVG with n_i and n_j as its children, and the messages sent by both nodes are used to refine $gP(e)$. Also notice that the S set calculation for a given e only makes use of $sA(n_i)$ and $sA(n_j)$ and not any other safe area, because those two safe areas summarize everything that might pass beyond them. The detailed procedure for calculating the S set is given by Algorithm 7.

Algorithm 7 Calculate the S set for a given evader position.

Input: RVG, set $\mathbf{V}$ that contains reflex vertices, set of safe areas sA, position of the evader e.
Output: S set.
if e is a node in RVG **then**
  1. Obtain node $n_i \in RVG$ equal to e;
  2. $S = sA(n_i)$;
else
  3. Initialize $S = gP(e)$;
  for every $v \in \mathbf{V}$ **do**
    4. $S \leftarrow S \cap GetMessageFromTo(v, e, sA)$;
  end for
end if
Return S;

In this game the evader is allowed to start in any point within the environment, lets call that initial position e_{init}. Then we connect its initial position to the RVG, tracing line segments from e_{init} to any node $n_i \in$ RVG, whenever it is possible. For considering such scenario within the pursuer's surveillance strategy, we consider e_{init} as a new node in the RVG (properly connected as mentioned about), then using

Algorithm 7 we can calculate an S set that considers that the evader is in e_{init}. From that initial conditions, the pursuer's surveillance strategy, as was exactly described in this section, is then used. Furthermore, if initially the pursuer lies over the S set related to e_{init}, then it will be able to keep itself within the S set at all time.

8 Decidability and complexity results

8.1 On the decidability of the problem

Along this section we do some theoretical analysis about the procedure presented in Sect. 6.2. We first present a modeling of the problem based on composition of relations (Takeuti and Zaring 1971) that facilitates the whole analysis, then using this modeling we present a proof of the algorithms' convergence, indeed, we also give a proof that demonstrates that the algorithms do converge in a finite number of iterations showing that the problem is decidable.

For the case when the RVG has a tree topology, as shown in Sect. 6.1, for the correct calculation of the safe areas only two tree traversals are needed. One from the leaf nodes all the way up to the root node, and the other, from the root node all the way down to the leaf nodes (see Theorem 1 proof). Hence, the safe areas are obtained with Algorithm 1 in a finite number of iterations.

On the other hand, when the RVG contains cycles intrinsic to its topology, some questions about the convergence of the presented algorithms arise. For such cases the proposed solution for calculating the required safe areas are Algorithm 6, and Subroutines 8–10. Such algorithm works in a iterative way performing a series of traversals on a spanning tree, built based on the original RVG. Such series of traversals correspond to the iterations of the "do while" loop in Subroutine 8 when called in the point where the cycle starts. Hence, the questions that we address in this section are the next ones: as the "do while loop" keeps iterating, do the proposed algorithms keep getting closer to the correct value of the safe areas? Is the "do while" loop able to converge to the correct value of the safe areas in a *finite* number of iterations? Or, as it keeps iterating, it keeps getting infinitely closer to the correct solution without actually reaching it in a finite number of iterations, in which case our problem might be undecidable.

The presented modeling is based on relations, composition of relations and their preimages. The key idea consists on representing the valid set of points for the pursuer to maintain surveillance as a preimage of a relation between guard polygons.

Let us consider without loss of generality a single cycle $q_1 \rightarrow q_2 \rightarrow \cdots \rightarrow q_{n-1} \rightarrow q_n$, where q_i may be either a reflex vertex or a transition-point, and where the evader first visits the point q_1 and returns to that point after traveling the loop, we mean $q_n = q_1$. Now, let us focus on a pair of points q_i and q_{i+1} within that loop. If the evader travels from q_i to q_{i+1} and the pursuer is able to maintain surveillance of it, then there should be a valid subset of departure points $D \subseteq gP(q_i)$ where the pursuer can be, which allows to it to reach on time a subset of points $H \subseteq gP(q_{i+1})$. Each point in D will be able to reach its own subset of points in H; such mapping can be modeled as the relation $\mathbf{R}_{i,i+1}$ in Eq. 3.[4] Let $Im^{-1}(\mathbf{R}_{i,i+1})$ denote the preimage of the relation $\mathbf{R}_{i,i+1}$. In terms of our problem, the preimage $Im^{-1}(\mathbf{R}_{i,i+1})$ is the set of valid departure points over $gP(q_i)$ such that when the

Subroutine 8 $ReceiveChildNodesMessagesC(n_i, sA)$

Input: $n_i \in RVG$ is the node that receives the message from its children, sA is the current set of safe areas.
Output: The modified set of safe areas sA.
repeat
  1. $past\_sA(n_i) \leftarrow sA(n_i)$;
  if $ReturnFrom(n_i)$ is $True$ **then**
    Return sA;
  else
    for every $n_j \in children(n_i)$ **do**
      A. $sA \leftarrow ReceiveChildNodesMessagesC(n_j, sA)$;
      B. $sA(n_i) \leftarrow sA(n_i) \cap GetMessageFromTo(n_j, n_i, sA)$;
    end for
  end if
  2. $sA(\,DescendantNodeEqualTo(n_i)\,) \leftarrow sA(n_i)$;
  3. $Iterations(n_i) \leftarrow Iterations(n_i) + 1$;
until $Stop(n_i, past\_sA(n_i), sA(n_i))$ is $True$
4. $Iterations(n_i) \leftarrow 0$;
Return sA;

Subroutine 9 $ReturnFrom(n_i)$

Input: $n_i \in RVG$.
Output: A $True$ of $False$ value that indicates to stop or not iterating through a cycle.
if n_i is a $leafNode$ **then**
  Return $True$;
else if $aCycleIsClosedIn(\,parent(n_i)\,)$ is $True$ $\bigwedge$ $0 < Iterations(\,AncestorNodeEqualTo(\,parent(n_i)\,)\,)$ **then**
  Return $True$;
else
  Return $False$;
end if

Subroutine 10 $Stop(n_i, past\_sA(n_i), sA(n_i))$

Input: $n_i \in RVG$, $past\_sA(n_i)$ is the past value of the safe area related to n_i, $sA(n_i)$ is the current value of the safe area related to n_i..
Output: A $True$ of $False$ value that indicates to stop or not iterating through a cycle.
if $past\_sA(n_i) \neq sA(n_i) \bigwedge aCycleBeginsIn(n_i)$ is $True$ **then**
  Return $False$;
else
  Return $True$;
end if

[4] $t_c(w, z)$ denotes the smallest time that takes to the player c to move from point w to point z.

pursuer starts from them, it is able to maintain surveillance of the evader when this last one visits the sequence $q_i \rightarrow q_{i+1}$.

$$\mathbf{R}_{i,i+1} = \Big\{(w,z) : t_p(w,z) \leq t_e(q_i,q_{i+1}) \quad \text{where } w \in gP(q_i) \text{ and } z \in gP(q_{i+1})\Big\} \tag{3}$$

If we consider a third point q_{i+2} so the evader now goes from q_i to q_{i+2} passing through q_{i+1}, then a point $w \in gP(q_i)$ is a valid departure point for the pursuer when it wants to maintain surveillance of the mentioned evader's path, if there exists a point $z \in gP(q_{i+2})$ such that $(w,y) \in \mathbf{R}_{i,i+1} \wedge (y,z) \in \mathbf{R}_{i+1,i+2}$, which fits the definition of a composition of relations. In other words, if the evader travels the sequence $q_i \rightarrow q_{i+1} \rightarrow q_{i+2}$, then a valid departure point w, which allows the pursuer to maintain evader's surveillance, must belong to the preimage $Im^{-1}(\mathbf{R}_{i+1,i+2} \circ \mathbf{R}_{i,i+1})$. Based on that reasoning let us denote $\mathbf{R}_{1,n} = \mathbf{R}_{n-1,n} \circ \mathbf{R}_{n-2,n-1} \circ \cdots \circ \mathbf{R}_{1,2}$, as the mapping process that suffers a pursuer's position $w \in gP(q_1)$ while the evader travels the cycle one time. Furthermore, we will denote $\mathbf{R}^N_{1,n} = \mathbf{R}_{1,n} \circ \cdots \circ \mathbf{R}_{1,n}$ with $N-1$ compositions, as the progressive pursuer's position mapping that considers that the evader travels the cycle N times. Actually, what it is done in each iteration k of the "do while" loop in Subroutine 8 when called at q_1, is to calculate the preimage $Im^{-1}(\mathbf{R}^k_{1,n})$ [which is $sA(q_1)^k$], starting with $k=1$, then $k=2$, etc., and as the algorithm keeps iterating, evader's traversals of more and more laps around the mentioned cycle are considered. Specifically, the preimage $Im^{-1}(\mathbf{R}^\infty_{1,n})$ is crucial, where

$$\mathbf{R}^\infty_{1,n} = \lim_{k \longrightarrow \infty} \mathbf{R}^k_{1,n}$$

In this pursuit-evasion problem, $Im^{-1}(\mathbf{R}^\infty_{1,n})$ is the set of valid departure points over $gP(q_1)$ that allow to the pursuer to maintain evader's surveillance if the evader decides to traverse any number of laps on the considered cycle (the case when the evader keeps going forever around that cycle is included). Notice that the preimage $Im^{-1}(\mathbf{R}^\infty_{1,n})$ is the safe area $sA(q_1)$ corresponding to an evader that just keeps traveling around such cycle in the predefined direction.

Once we have presented a proper modeling using composition of relations, we will proceed to prove through Theorem 2 the convergence of Algorithm 6 to the desired solution. Notice that Theorem 2 claims convergence but not in a finite number of iterations, which is then proved in Theorem 3.

Theorem 2 *Consider an evader that is continuously traveling across a path $q_1 \rightarrow q_2 \rightarrow \cdots \rightarrow q_{n-1} \rightarrow q_1$, making loops around that cycle. As Algorithm 6 keeps iterating, the calculated preimage $Im^{-1}(\mathbf{R}^k_{1,n})$ converges to $Im^{-1}(\mathbf{R}^\infty_{1,n})$, which is the safe area $sA(q_1)$ that corresponds to that evader's behavior.*

Proof First of all, we know that any further composition $\mathbf{R}_{1,n}$ to $\mathbf{R}^\infty_{1,n}$ does not alter its preimage, therefore, $Im^{-1}(\mathbf{R}_{1,n} \circ \mathbf{R}^\infty_{1,n}) = Im^{-1}(\mathbf{R}^\infty_{1,n})$ which can be rewritten as:

$$\begin{aligned} Im^{-1}\big(\mathbf{R}^{N+1}_{1,n}\big) &\longrightarrow Im^{-1}\big(\mathbf{R}^N_{1,n}\big) \quad \text{for } N \longrightarrow \infty \\ Im^{-1}\big(\mathbf{R}^N_{1,n}\big) \backslash Im^{-1}\big(\mathbf{R}^{N+1}_{1,n}\big) &\longrightarrow \emptyset \quad \text{for } N \longrightarrow \infty \end{aligned} \tag{4}$$

Hence, as Algorithm 6 keeps iterating, the preimage $Im^{-1}(\mathbf{R}^k_{1,n})$ calculated by such algorithm at each iteration k must have the next property:

$$\lim_{k \longrightarrow \infty} Im^{-1}\big(\mathbf{R}^k_{1,n}\big) \setminus Im^{-1}\big(\mathbf{R}^{k+1}_{1,n}\big) = \emptyset \tag{5}$$

We also know that the relations composition that considers k evader's laps to the cycle can be written as $\mathbf{R}^{k+1}_{1,n} = \mathbf{R}_{1,n} \circ \mathbf{R}^k_{1,n}$, where $\mathbf{R}^k_{1,n}$ is the inner relation that will be further composed through the outer relation $\mathbf{R}_{1,n}$. Using a standard methodology to calculate $Im^{-1}(\mathbf{R}^{k+1}_{1,n})$, we first take the preimage of the inner relation, $Im^{-1}(\mathbf{R}^k_{1,n})$, as a first approach of $Im^{-1}(\mathbf{R}^{k+1}_{1,n})$. Subsequently, by removing the correct elements from $Im^{-1}(\mathbf{R}^k_{1,n})$, we add the restrictions given by the values of $Im(\mathbf{R}^k_{1,n})$ for which $Im^{-1}(\mathbf{R}_{1,n})$ is not defined (see Fig. 9). Hence we can conclude that:

$$Im^{-1}\big(\mathbf{R}^{k+1}_{1,n}\big) \subseteq Im^{-1}\big(\mathbf{R}^k_{1,n}\big). \tag{6}$$

Based on Eq. 6 two possibilities arise. The first possibility is that the equality in Eq. 6 is fulfilled, in which case the condition given by Eq. 5 is also fulfilled telling us that Algorithm 6 has already converged. This past result can be obtained as follows. Let us suppose that the equality in Eq. 6 is fulfilled at iteration $s+1$, namely, $Im^{-1}(\mathbf{R}^{s+1}_{1,n}) = Im^{-1}(\mathbf{R}^s_{1,n})$. From this point, for any iteration $k \geq s$ we would obtain $Im^{-1}(\mathbf{R}^{k+1}_{1,n}) = Im^{-1}(\mathbf{R}^k_{1,n})$; this equality is proved by induction. As the induction step we have:

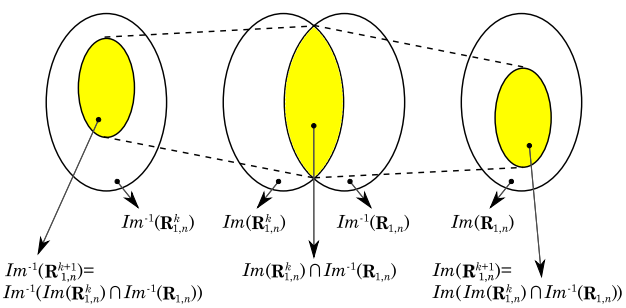


Fig. 9 Structure of the composition $\mathbf{R}^{k+1}_{1,n} = \mathbf{R}_{1,n} \circ \mathbf{R}^k_{1,n}$

$Im^{-1}(\mathbf{R}^{k+1}_{1,n}) = Im^{-1}(Im(\mathbf{R}_{1,n}) \cap Im^{-1}(\mathbf{R}^{k}_{1,n}))$, and assuming $Im^{-1}(\mathbf{R}^{k}_{1,n}) = Im^{-1}(\mathbf{R}^{k-1}_{1,n})$, then $Im^{-1}(\mathbf{R}^{k+1}_{1,n}) = Im^{-1}(Im(\mathbf{R}_{1,n}) \cap Im^{-1}(\mathbf{R}^{k-1}_{1,n}))$. We also know that $Im^{-1}(\mathbf{R}^{k}_{1,n})$ is equal to $Im^{-1}(Im(\mathbf{R}_{1,n}) \cap Im^{-1}(\mathbf{R}^{k-1}_{1,n}))$, hence, we conclude that $Im^{-1}(\mathbf{R}^{k+1}_{1,n}) = Im^{-1}(\mathbf{R}^{k}_{1,n})$ using $Im^{-1}(\mathbf{R}^{s+1}_{1,n}) = Im^{-1}(\mathbf{R}^{s}_{1,n})$ as the base case. So, if $Im^{-1}(\mathbf{R}^{k}_{1,n})$ does not have any further change from $k = s$, then as $k \longrightarrow \infty$, we have $Im^{-1}(\mathbf{R}^{k}_{1,n}) \setminus Im^{-1}(\mathbf{R}^{k+1}_{1,n}) = \emptyset$, which is actually Eq. 5, telling us that at iteration $k = s$ the convergence has already been achieved. The result follows, precisely, for the first possibility when the equality in Eq. 6 is fulfilled.

The second possibility is that the equality in Eq. 6 is not fulfilled in which case we have:

$$\ldots Im^{-1}\left(\mathbf{R}^{k+1}_{1,n}\right) \subset Im^{-1}\left(\mathbf{R}^{k}_{1,n}\right) \subset Im^{-1}\left(\mathbf{R}^{k-1}_{1,n}\right) \ldots \quad (7)$$

Using Eq. 7, if $Im^{-1}(\mathbf{R}^{k+1}_{1,n}) \subset Im^{-1}(\mathbf{R}^{k}_{1,n})$ it implies that $Im^{-1}(\mathbf{R}^{k}_{1,n}) \setminus Im^{-1}(\mathbf{R}^{k+1}_{1,n}) \subset Im^{-1}(\mathbf{R}^{k}_{1,n})$. As a consequence, $Im^{-1}(\mathbf{R}^{k}_{1,n})$ can be treated as an upper bound to $Im^{-1}(\mathbf{R}^{k}_{1,n}) \setminus Im^{-1}(\mathbf{R}^{k+1}_{1,n})$. On the other hand, we know that $Im^{-1}(\mathbf{R}^{k}_{1,n})$ cannot be smaller than $Im^{-1}(\mathbf{R}^{\infty}_{1,n})$, so $Im^{-1}(\mathbf{R}^{k}_{1,n}) \setminus Im^{-1}(\mathbf{R}^{k+1}_{1,n})$ has the empty set $\emptyset$ as a lower bound. Now, due to Eq. 7 we can observe that the upper bound $Im^{-1}(\mathbf{R}^{k}_{1,n})$ is contracting as the iteration index k increases, pushing $Im^{-1}(\mathbf{R}^{k}_{1,n}) \setminus Im^{-1}(\mathbf{R}^{k+1}_{1,n})$ at each iteration to its lower bound $\emptyset$, hence as the iterations of Subroutine 8 when called at q_1 tend to infinity, $Im^{-1}(\mathbf{R}^{k}_{1,n}) \setminus Im^{-1}(\mathbf{R}^{k+1}_{1,n})$ tends to $\emptyset$, which actually is the property given by Eq. 5. The result follows. □

Concerning the decidability of our problem, the question that might arise is whether the contraction presented in Eq. 7 keeps going forever such that Algorithm 6 and its subroutines are not able to calculate $Im^{-1}(\mathbf{R}^{\infty}_{1,n})$ in finite time, which might suggest that the problem is undecidable. However, that is not the case, in Theorem 3 (below) we will prove that our surveillance problem is decidable. As an outline of the proof of Theorem 3, we first assume that we have already calculated $Im^{-1}(\mathbf{R}^{1}_{1,n})$ (actually $\mathbf{R}^{1}_{1,n}$ is $\mathbf{R}_{1,n}$), then what we do is to base our analysis on the number of laps that the pursuer can maintain surveillance of the evader. That number of laps is mapped into the plane generating two groups within $Im^{-1}(\mathbf{R}^{1}_{1,n})$. The first group is formed by points over $Im^{-1}(\mathbf{R}^{1}_{1,n})$ that will be valid departure points for the pursuer such that if it starts from them it will be able to maintain evader's surveillance during any number of laps, in other words, it can watch the evader even if it decides to travel an "*infinite*" number of laps. The second group in $Im^{-1}(\mathbf{R}^{1}_{1,n})$, are valid departure points for the pursuer such that if it starts from them it will just be able to maintain evader's surveillance during a *finite* number of laps. Finally, analyzing the different cases whether the first or second group are empty sets, we establish the decidability result.

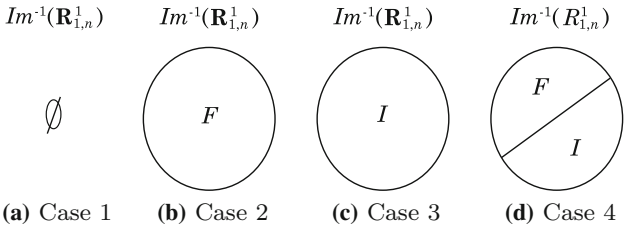


Fig. 10 Venn diagrams of the decompositions of $Im^{-1}(\mathbf{R}^{1}_{1,n})$ presented in the cases of Theorem 3

Theorem 3 *The proposed algorithm always converges in a finite number of iterations, hence, the problem of deciding whether or not a pursuer is able to maintain SMV of an evader that travels over the RVG, both players moving at bounded speed, is decidable.*

Proof Consider an evader that is continuously traveling across a path $q_1 \rightarrow q_2 \rightarrow \cdots \rightarrow q_{n-1} \rightarrow q_1$, making loops around that cycle. Also consider that we have run one iteration of the "do while" loop in Subroutine 8 calling it in q_1, such that $Im^{-1}(\mathbf{R}^{1}_{1,n})$ has already been calculated, that is, we already have calculated the departure points over $gP(q_1)$ that allow the pursuer to maintain SMV of an evader for at least one lap. Now, we proceed to partition $Im^{-1}(\mathbf{R}^{1}_{1,n})$ into two sets. The first one, let us call it I, which will be the set of points over $Im^{-1}(\mathbf{R}^{1}_{1,n})$ that are valid departure points for the pursuer such that they allow it to maintain SMV of an evader that travels an "*infinite*" number of laps around the cycle, which means that $I = Im^{-1}(\mathbf{R}^{\infty}_{1,n})$. The second set, lets call it F, will be the set of points in $Im^{-1}(\mathbf{R}^{1}_{1,n})$ such that when the pursuer departs from them, they *just* allow the pursuer to maintain SMV of the evader during a *finite* number of laps around the cycle. We mean that $Im^{-1}(\mathbf{R}^{1}_{1,n}) \setminus Im^{-1}(\mathbf{R}^{k}_{1,n}) \in F$,[5] for $k = 2, 3, 4, \ldots, M+1$, with M being a natural number denoting the maximum finite number of laps that the pursuer can maintain evader's surveillance. We can think in F as the complement of I in $Im^{-1}(\mathbf{R}^{1}_{1,n})$, that is, $F = Im^{-1}(\mathbf{R}^{1}_{1,n}) \setminus I$.

Based on such partition of $Im^{-1}(\mathbf{R}^{1}_{1,n})$ four cases may arise considering whether or not, either I or F are empty sets (see Fig. 10). The four cases are the next ones:

– *Case 1* $I = \emptyset$ and $F = \emptyset$. Due to the fact that $Im^{-1}(\mathbf{R}^{1}_{1,n}) = I \cup F$, this case implies that $Im^{-1}(\mathbf{R}^{1}_{1,n})$ is equal to $\emptyset$. This means that from the very first iteration of Subroutine 8, we notice that there are no departure

[5] The set $Im^{-1}(\mathbf{R}^{1}_{1,n}) \setminus Im^{-1}(\mathbf{R}^{k}_{1,n})$ is formed by the elements in $Im^{-1}(\mathbf{R}^{1}_{1,n})$ such that if the pursuer departs from them, it can maintain SMV of the evader for *at most* $k-1$ laps.

points for the pursuer in $gP(q_1)$ that allow it to maintain SMV of the evader, not even for one lap through the cycle. In this case the evader wins.

- *Case 2* $I = \emptyset$ and $F \neq \emptyset$. As we already know, when Subroutine 8 is called from q_1, each iteration k of the "do while" loop calculates the preimage $Im^{-1}(\mathbf{R}^k_{1,n})$. Note that at iteration k the elements $Im^{-1}(\mathbf{R}^1_{1,n}) \setminus Im^{-1}(\mathbf{R}^k_{1,n})$ are also being removed from $Im^{-1}(\mathbf{R}^1_{1,n})$, which are all the departure points for the pursuer that *just* allow it to maintain surveillance of the evader during $k-1$ or less laps to the cycle. By the definition of the set F, we know that it contains valid pursuer's departure points that allow evader's surveillance for at most M laps (recall M is a natural number), which means that in the iteration $k = M+1$ the whole set F must have been removed, and taking into account that $I = \emptyset$, at iteration $k = M+1$ we have $Im^{-1}(\mathbf{R}^k_{1,n}) = \emptyset$ allowing us to determine that the evader just needs to take $M+1$ laps to win the game. Above all, notice that the number $M+1$ can be huge but at the end it is reachable in finite time. Also notice that in each iteration $k \leq M+1$ there must be a continuous removal of points from $Im^{-1}(\mathbf{R}^1_{1,n})$, because if somehow in an iteration $k \leq M+1$ there is no removal of points, we have $Im^{-1}(\mathbf{R}^k_{1,n}) = Im^{-1}(\mathbf{R}^{k-1}_{1,n})$ that as seen in Theorem 2's proof, that causes $Im^{-1}(\mathbf{R}^k_{1,n})$ not have any further change indicating the existence of $Im^{-1}(\mathbf{R}^\infty_{1,n})$ within $Im^{-1}(\mathbf{R}^k_{1,n})$, which contradicts the assumptions of this case that is $I = \emptyset$.
- *Case 3* $I \neq \emptyset$ and $F = \emptyset$. We obtained $Im^{-1}(\mathbf{R}^1_{1,n})$ when a first iteration of Subroutine 8 is called from q_1. Due to the fact that $F = \emptyset$, $Im^{-1}(\mathbf{R}^1_{1,n})$ only contains $I = Im^{-1}(\mathbf{R}^\infty_{1,n})$. If we execute a second iteration, that is $k = 2$, we obtain that $Im^{-1}(\mathbf{R}^2_{1,n}) = Im^{-1}(\mathbf{R}^1_{1,n})$. Furthermore, we obtain $Im^{-1}(\mathbf{R}^{k+1}_{1,n}) = Im^{-1}(\mathbf{R}^k_{1,n})$ for any iteration $k \geq 1$, as stated by the argument presented in Theorem 2's proof. This means that once we have calculated $Im^{-1}(\mathbf{R}^1_{1,n})$, it will never change in any further iteration, telling us that actually $Im^{-1}(\mathbf{R}^1_{1,n}) = Im^{-1}(\mathbf{R}^\infty_{1,n})$. Due to such arguments, we can say that in iteration $k = 2$ if we notice that $Im^{-1}(\mathbf{R}^2_{1,n}) = Im^{-1}(\mathbf{R}^1_{1,n})$, then we can be sure that we have already calculated $Im^{-1}(\mathbf{R}^\infty_{1,n})$, which means that the pursuer wins because it is able to maintain SMV of an evader that takes any number of laps around the cycle.
- *Case 4* $I \neq \emptyset$ and $F \neq \emptyset$. Similarly as it happened in case 2, at iteration $k = M+1$ the whole set F must have been removed because by definition it contains valid pursuer's departure points that allow evader's surveillance for at most M laps. Hence, at iteration $k = M+1$ $Im^{-1}(\mathbf{R}^k_{1,n})$ only contains the set $I = Im^{-1}(\mathbf{R}^\infty_{1,n})$, so as happened in case 3, we have $Im^{-1}(\mathbf{R}^{k+1}_{1,n}) = Im^{-1}(\mathbf{R}^k_{1,n})$ for any $k \geq M+1$, telling that $Im^{-1}(\mathbf{R}^\infty_{1,n})$ has already been computed. Furthermore, at iteration $k = M+2$ we are able to tell that $Im^{-1}(\mathbf{R}^\infty_{1,n})$ exists concluding that the pursuer wins because it is able to maintain SMV of an evader that takes any number of laps around the cycle.

Through the past four cases it can be seen that in any situation we are able to determine if the pursuer is able to maintain surveillance of the evader in a *finite* number of iterations that is at must $M+2$ iterations, which indicates that the problem is *decidable*. The result follows. □

Remark 4 The problem of deciding which player wins is history dependent. It is not possible to decide which player wins considering only pairs of relations independently.

8.2 A complexity result

Recall that any instance of our decision problem stated in Sect. 4 is given by the tuple $(V, A, \{R_i\}, V_{ratio}, \Lambda, \Pi)$, where V and A represent respectively the nodes and the edges of the RVG, that is $RVG = (V, A)$, $\{R_i\}$ is the partition of the environment, $V_{ratio} = V_p/V_e$ is the speed ratio of the players, Λ is the size of the paths that the evader travels in terms of the number of critical points that it visits, and Π the set of particular properties that define the evader's paths.

The evader moves between critical points, while the pursuer moves between guard polygons. Whether or not the pursuer or evader has a winning strategy for a given instance $(V, E, \{R_i\}, V_{ratio}, \Lambda, \Pi)$ of our problem, amounts to check the cost over the edges connecting critical points for any evader's path that fulfills the size constraint Λ and the set of properties Π. The cost associated with every edge is simply given by $t^k_p - t^k_e$, where t^k_e be the time that takes to the evader to reach the critical point $k+1$ from its k critical point, and t^k_p the time that takes the pursuer to reach its corresponding $k+1$ guard polygon departing from the k guard polygon.

The evader has a winning strategy if there exists a path for which the constraint $t^k_p - t^k_e \leq 0$ is broken. Conversely, the pursuer has a winning strategy if for every path it preserves the constraint. Notice that all the paths cost must be checked to reach a decision. This is the rationale behind the proof of Theorem 4.

Notice that the algorithms presented on the past sections were built over the most general instances of our problem, where the RVG may have cycles or not, for any partition $\{R_i\}$ and any V_{ratio} (we might have players with equal velocities or either player might be faster), and where Λ might tend to ∞ and $\Pi = \emptyset$; hence, we considered evader paths of any size with no particular restriction over them. If Λ is set to a particular value and $\Pi \neq \emptyset$, the message passing scheme is maintained but applied to the evader paths that fulfill Λ and Π. The cost over the edges connecting critical points can be

abstracted to the existence of safe areas obtained by Sect. 6 algorithms. *If and only if a safe area is empty then the cost of moving among critical points in the partition has a positive cost.*

The problem of deciding which player wins is NP-complete as we will prove later on this section. The decision version of *bottleneck traveling salesman* problem (TSP), known to be NP-complete (Garey and Johnson 1979), has an immediate equivalence to some specific instances of the problem of establishing a feasibility cost over the edges connecting critical points.

The decision version of the bottleneck TSP is defined as following:

Definition 4 For a given length x, is there a Hamiltonian cycle in a graph g with no edge longer than x? (Garey and Johnson 1979).

The proof consists on a reduction by restriction (Garey and Johnson 1979). What it is done is to add restrictions to the instances of the problem, which results into a restricted problem that is identical to the decision version of the bottleneck TSP. In other words, we show that our decision problem contains the bottleneck TSP as an special case. The restricted problem consists in an evader traveling Hamiltonian cycles, which are the paths that consider visiting all the different nodes in the RVG. This is one type of paths among all the possible paths that the evader might try to escape.

Theorem 4 *The problem of deciding whether or not the pursuer is able to maintain SMV of an evader that travels over the RVG, both players moving at bounded speed, is NP-complete.*

Proof In order to make a formal reduction, we use the concept of critical points and guard regions. Let us consider environments in which the critical points are only reflex vertices. Those reflex vertices are the nodes of the RVG. The evader moves between reflex vertices, while the pursuer moves between the corresponding guard polygons. The cost related to the RVG edges is given by $t_p^k - t_e^k$, where t_e^k is the time that takes to the evader to reach the reflex vertex $k+1$ from its k reflex vertex, and t_p^k the time that takes the pursuer to reach its corresponding $k+1$ guard polygon departing from the k guard polygon.

We assume, without loss of generality, that both players move at saturated speed. Let us also consider I_{HC} as the set of instances of our decision problem that are obtained by making Π contain the necessary properties such that the evader's path are Hamiltonian cycles, that is, the paths followed by the evader will be permutations of nodes $\langle n_1, n_2, \ldots, n_{|V|}, n_{|V|+1}\rangle$, which automatically sets $\Lambda = |V|+1$, where n_i and n_{i+1} are connected in $RVG = (V, A)$, and where $n_i \neq n_j$, except for $n_{|V|+1}$ which is equal to n_1.

The reduction consists in defining the cost over the RVG edges in the instances of I_{HC}, as the edge weights of the bottleneck traveling salesman problem. Answering whether or not there is a Hamiltonian cycle in the RVG with no edge longer than C will both solve: the decision version of the bottleneck traveling salesman problem and the instances in I_{HC} of deciding which player will win. A pursuer solution corresponds to $C \leq 0$. A polynomial time algorithm capable of solving these instances of our decision problem would also solve all instances of the decision version of the bottleneck traveling salesman problem in polynomial time. Therefore, our decision problem is NP-complete. □

Notice that the complexity result in Theorem 4 holds regardless of the algorithm used to solve the proposed problem, since the evader is free to choose any problem instance, for example, it can choose to travel a specific one that purely include Hamiltonian paths, which are computational costly.

Remark 5 The issues of considering environments with transition points or evader motions over repeated cycles are not needed to establish the complexity of this problem. These issues can only increase the complexity.

Remark 6 To get around the need of a computationally infeasible algorithm, the players might use an on-line strategy. An on-line strategy computes a motion plan for the next h future stages, and re-plans in the next iteration for the following h future stages. Typically, h is a small number.

9 Simulations results

On this section we present some simulation results, making use of implementations of the proposed algorithms. The objective of these simulations is to graphically show the safe areas and S sets under different conditions, specifically, the topology of the map (with or without holes), the topology of the RVG (with or without cycles), and different velocity ratios of the players (pursuer faster than the evader or vice-versa). All our simulation experiments were run on a dual core processor PC, equipped with 1 GB of RAM, running Linux, and were programmed in C++ using the computational geometry library LEDA.

In the next figures we present in dark grey (green) the computed safe areas. In Fig. 11 we show a sample environment with its partition, the resulting safe areas and the respective RVG. Something interesting to note in this example is that the calculations consider a pursuer that moves at a 90 % of the evader's velocity ($V_p/V_e = 0.9$), resulting into safe areas that are not empty sets, showing that there exists maps where an slower pursuer can maintain surveillance of a faster evader. The safe areas along with the environment partition and the necessary graphs, were computed in 0.2 s.

Figure 12 shows a more complex map with a tree topology RVG. In this example we considered a pursuer 10 % faster

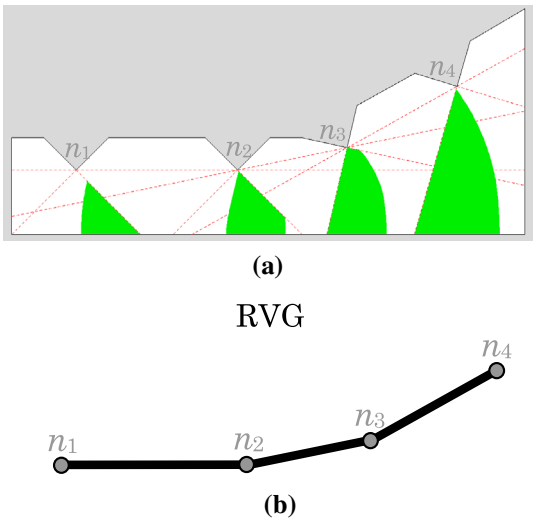


Fig. 11 Example 1 with a tree topology RVG and its calculated safe areas

than the evader ($V_p/V_e = 1.1$). For a better appreciation of the resulting safe areas, Fig. 12c shows all of them separately; in that figure we can notice that the safe areas may result into curved polygons, whose forms are not precisely intuitive. Also notice that when the safe areas result into sets greater than a single point, there is a whole family of paths that the pursuer can choose from to move between safe areas. The environment partition, the necessary graphs and the safe areas were computed in 0.7 s.

Figure 13 shows an environment whose RVG has a cycle in its topology. It is worthwhile to mention that this map is not simply connected, showing in that way the validity of our solution for both simply and non-simply connected environments. In Fig. 13a we considered a pursuer 20 % faster than the evader ($V_p/V_e = 1.2$); keep in mind that it is impossible for a slower pursuer to maintain surveillance of a faster evader in an environment *with holes*. Actually in that case the safe areas result into empty sets. Such behaviour is shown in Fig. 13a, and c–e. In Fig. 13c a velocity ratio $V_p/V_e = 1.05$ is used, then in Fig. 13d the velocity ratio is set to 1.015, showing the fact that as the velocity ratio $V_p/V_e \to 1$, the safe areas tend to be single points giving just one possible path for the pursuer to move between safe areas, and when we have $V_p/V_e < 1$ the safe areas result into empty sets, such as depicted in Fig. 13e where a V_p/V_e was set to 0.999. Making use of Eq. 6, as stopping criteria of our algorithm we compare the area of $sA(n_r)^{(k)}$ between consecutive iterations, and once we have no further change on such area we stop iterating (that area was calculated with a double floating point precision). It is worthwhile to mention that what we present here are just computational geometry approximations, but for obtaining exact mathematical results that perfectly fit the theoretical results on the previous sections,

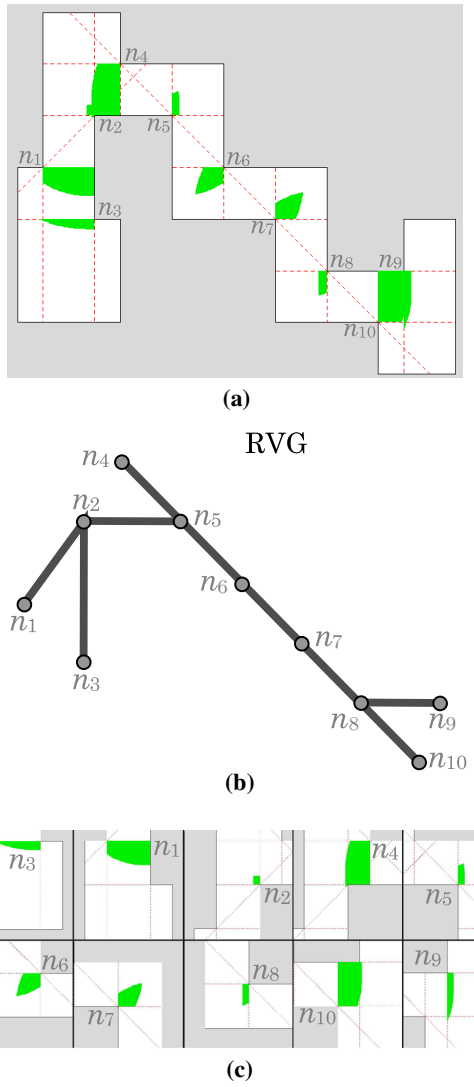
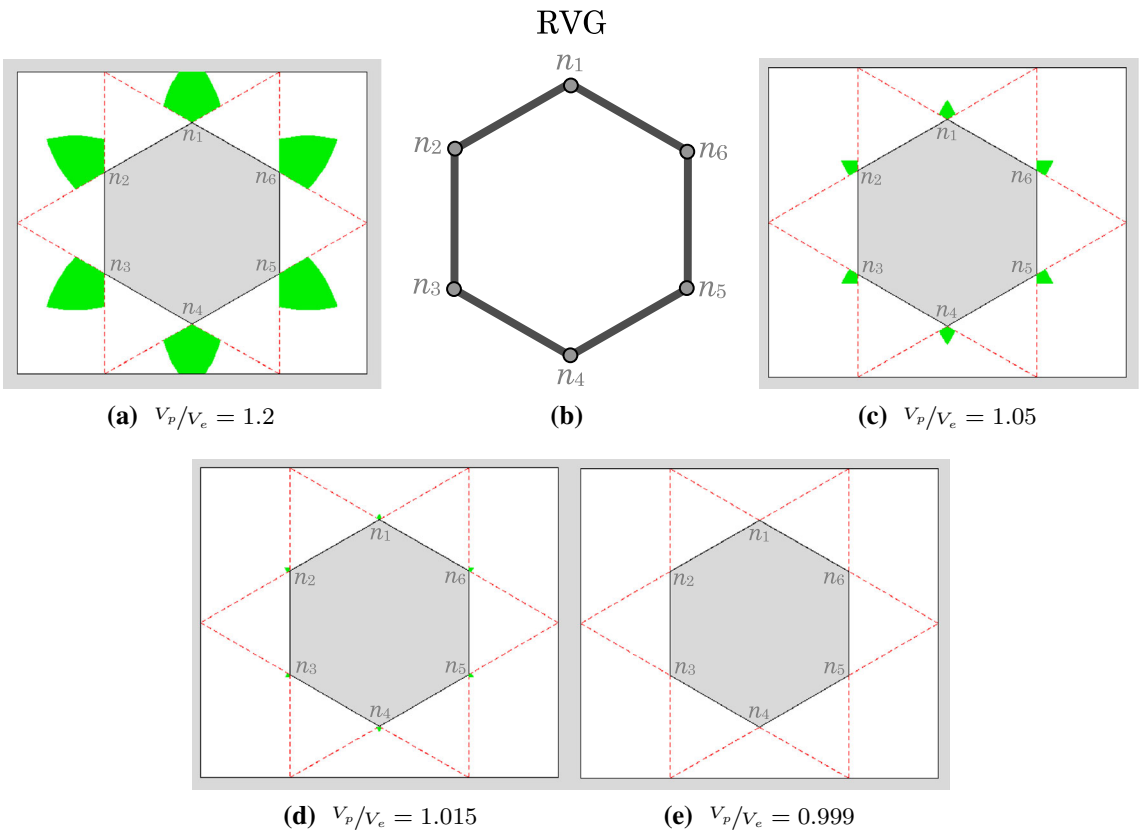


Fig. 12 Example 2 with a tree topology RVG and its calculated safe areas

a comparison between the algebraic expressions (which are line segments and circle arcs) that describe the boundaries of $sA(n_r)^{(k)}$ between consecutive iterations could be used as stopping criteria; however, that would imply that we need to implement algebraic simulations whose complexity go far beyond the illustration purposes intended by these simulations.

To conclude this discussion, when we considered $V_p/V_e = 1.2$ (Fig. 13a) the convergence was achieved in 7 iterations, for $V_p/V_e = 1.05$ (Fig. 13c) 9 iterations were needed, for $V_p/V_e = 1.015$ (Fig. 13d) 10 iterations, and for $V_p/V_e = 0.999$ (Fig. 13e) 2 iterations (recall that at least 2 iterations of the algorithm are required). The simulation that took the longest computation time was the one that considered the velocity ratio $V_p/V_e = 1.015$, and took 3.61 s.



(a) $V_p/V_e = 1.2$ **(b)** **(c)** $V_p/V_e = 1.05$

(d) $V_p/V_e = 1.015$ **(e)** $V_p/V_e = 0.999$

Fig. 13 Example 3 with cycles in the RVG and its calculated safe areas

In Figs. 14 and 15 we present a fourth and fifth experiment. The purpose behind these experiments is to evaluate how the execution time of the proposed algorithm increases as the number of reflex vertices also increases. The maps presented in Fig. 14 correspond to an expansion of the map presented in Fig. 12. The map in Fig. 14a was obtained doubling the number of reflex vertices of the map in Fig. 12a, yielding a total of 20 reflex vertices. The map in Fig. 14b triplicates the number of reflex vertices of the map in Fig. 12a for a total of 30 reflex vertices. In both maps the RVG has a tree topology. The safe areas in the map of 20 vertices where computed in 1.84 s, and in the map with 30 vertices they were computed in 3.16 s. The maps in Fig. 15 were constructed based on Fig. 13a. The map in Fig. 15a was obtained doubling the number of reflex vertices of the map in Figs. 13a, and 14b tripling such number of vertices, yielding, respectively, 12 and 18 reflex vertices. In both maps the RVG has cycles; also notice that the complexity of the maps increased, as the number of holes, hence, the number of intrinsic cycles, also increased. The safe areas in the map with 12 vertices where computed in 13.04 s, and in the map with 18 vertices they were computed in 31.69 s. In Fig. 16 a plot is presented, showing the variation of the execution time of the algorithm as the number of vertices increases. In Fig. 16 the x axis reflects the fact that the number of reflex vertices in Figs. 12a and 13a is double and tripled. The y axis shows the factor by which the execution time increased. In Fig. 16 it can be noticed that despite the fact that the number of vertices is increased by the same factor, the execution time increases faster in Example 5 (maps in Fig. 15). This shows that the number of vertices is not the only element that affects the execution time of the algorithm, the topology of the RVG also has an influence on it. A more detailed analysis of this issue is an interesting direction for future work.

In Fig. 17 we introduce the S set, which we show in light gray (yellow). As we already mentioned, the S set will be primarily a function of the evader's position, which we show as a red square. In Fig. 17a the evader is standing on the middle of the edge that joins the reflex vertices n_1 and n_2. In that image the S set is calculated generating reachable areas around the safe areas $sA(n_1)$ and $sA(n_2)$. We only make use of those two safe areas because they already summarize all the possible subsequent visits of the evader to any other reflex vertex in the RVG. For obtaining those results we consider a velocity ratio $V_p/V_e = 0.9$, while in Fig. 17b the velocity ratio is set to 0.78, and in Fig. 17c is set to 0.71, showing the fact that as the pursuer is slower the S set gets smaller giving less flexibility of movement to the pursuer (recall that the pursuer must maintain itself inside the S set at all time). Actually there is minimum value for the velocity ratio that

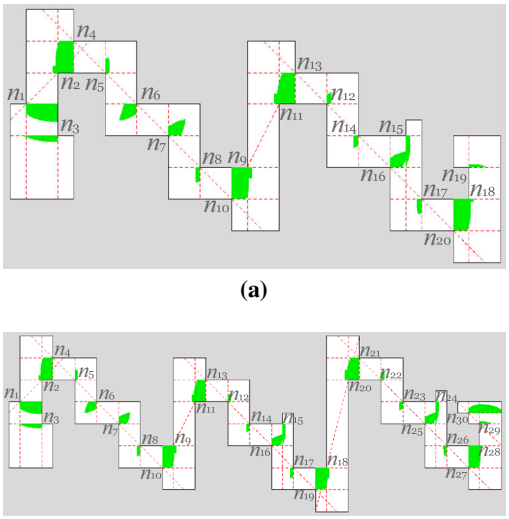


Fig. 14 Example 4 with tree topology RVG's and their calculated safe areas

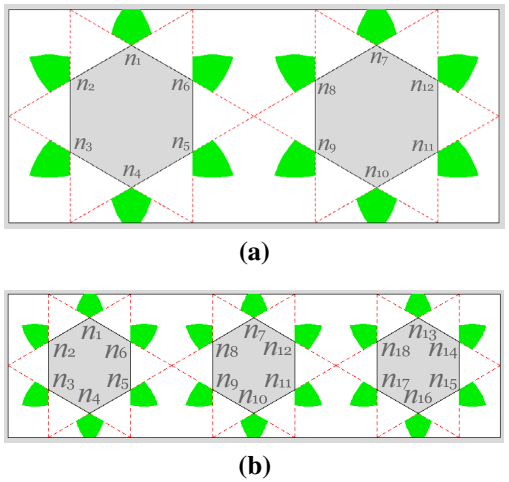


Fig. 15 Example 5 with cycles in the RVG's and their calculated safe areas

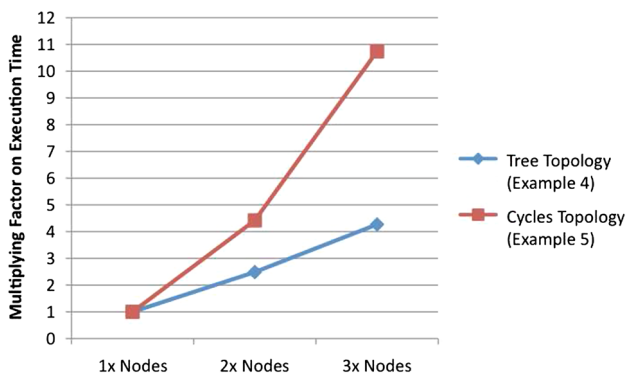


Fig. 16 Comparison chart showing how many times the execution time of the proposed algorithm increases as the number of vertices is doubled and tripled (Examples 4, 5)

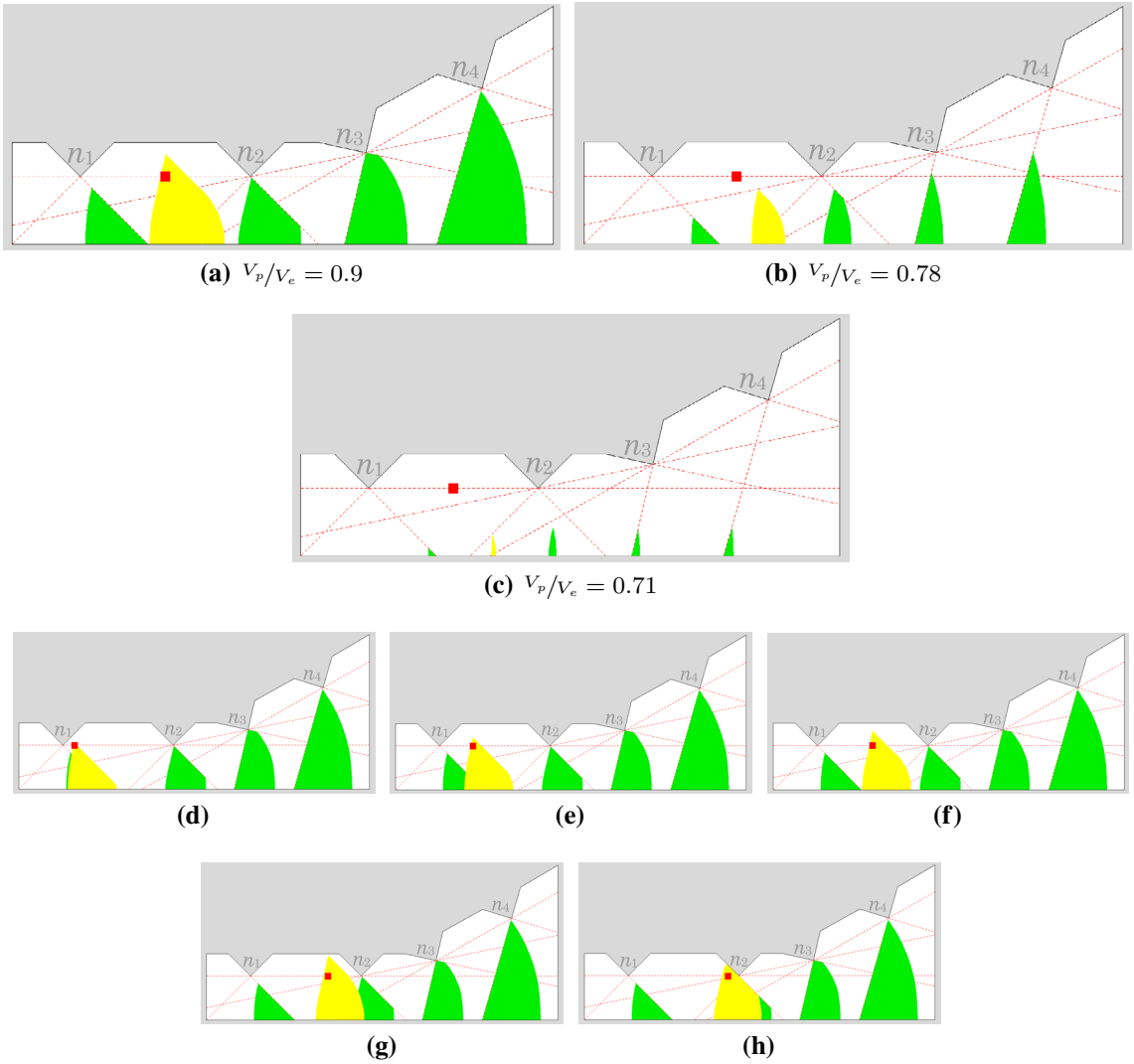
still allows to the pursuer to maintain evader surveillance; for smaller values of that minimum threshold the safe areas along with the S set collapse into empty sets. Figure 17d–h show how the S set changes and moves as the evader travel from n_1 to n_2. In average the S sets were calculated in 0.19 s.

It is worthwhile to mention that for all shown examples the complete solution was computed due to fact that no special restriction was considered on the paths traveled by the evader in the RVG. Nevertheless, keep in mind that since the evader is free to choose any path in the RVG, it can travel computational costly paths such as Hamiltonian paths, which yields the complexity result for the proposed problem regardless of the algorithm used to solve such problem. However, a solution that considers just a finite ahead horizon can always be computed. In Fig. 18 the same map of Fig. 11a is considered, but this time, the safe areas where computed considering just one step ahead, which results into bigger safe areas as they include only the restriction imposed by the next reflex vertex in the evader's path. The safe areas can always be computed off-line in a pre-computation stage. On the other hand, the simulation results also show that the S set can be computed on-line while the players move around the environment, as we were able to calculate them at a rate of 10 hertz, but we strongly believe that that rate can be improved with better hardware and implementing some computations in parallel (in the S set calculation, the messages for each $v \in \mathbf{V} = \{n_i, n_j\}$ sent to e can be computed in parallel).

10 Conclusion and future work

In this paper, we have addressed the problem of maintaining SMV of a moving evader by a pursuer, in an environment with obstacles, where the speed of each participant is bounded and there is not an a priory limit as to the distance that may separate them. More specifically we have addressed the problem of maintaining strong mutual visibility of a moving evader traveling in the shortest-path road-map (RVG), but under the assumption that the pursuer does not know, which among the possible paths, the evader will choose.

Central to our proposed solution is the concept of safe areas, that is, areas where the pursuer must be to keep evader surveillance at all time. Using preimage and compositions of relations we have shown that our algorithm will always converge in finite time, even when the evader decides to travel in a loop forever. If there is a solution for the pursuer, we have also found, the part of the space where the pursuer must be at every instant of time to guarantee evaders surveillance. Furthermore, based on the map only, and regardless of the initial position of the players, we can determine when the pursuer will not be able to keep SMV of the evader. We have implemented all our algorithms and presented simulation results.



(a) $V_p/V_e = 0.9$ **(b)** $V_p/V_e = 0.78$

(c) $V_p/V_e = 0.71$

(d) **(e)** **(f)**

(g) **(h)**

Fig. 17 Example 4 with its calculated safe areas and sample S sets

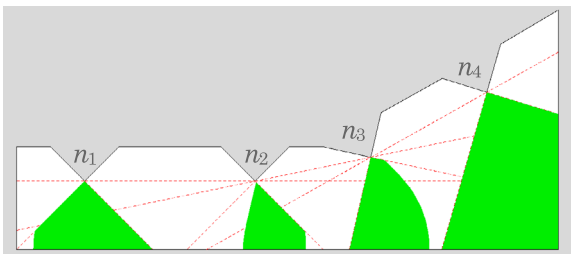


Fig. 18 Example 6 with safe areas calculated one step ahead

In short, we have provided two main contributions: (1) we proved decidability of this problem for any arbitrary polygonal environment, and (2) we provided a complexity measure to our evader surveillance game that holds regardless of the algorithm used to solve the game.

As future work, a possibility is to analyze in detail the case of an evader traveling over other roadmaps apart from the RVG, nevertheless, we would prefer to address the problem of maintaining SMV of a moving evader, in which the evader is free to travel any path within the workspace and the pursuer does not know the motion evader policy. A first effort in this direction is presented in Becerra et al. (2010). Moreover, we believe that we will be able to use our current progress as an stepping stone to solve that more general problem, since in an antagonist game a worst case scenario is commonly used to establish the solution of the game, and the RVG seems to be at least part of the worst case scenario since it contains the shortest paths to escape.

We would also like to consider the case, in which both the pursuer and the evader have bounded sensor range and bounded speed, and are disc shaped robots. We think that the concept of strong mutual visibility is specially suitable for the later case, since it defines visibility between regions with area.

Acknowledgments This work was partially funded by CONACYT Project 106475 and NSF-CONACYT Project J110.534/2006.

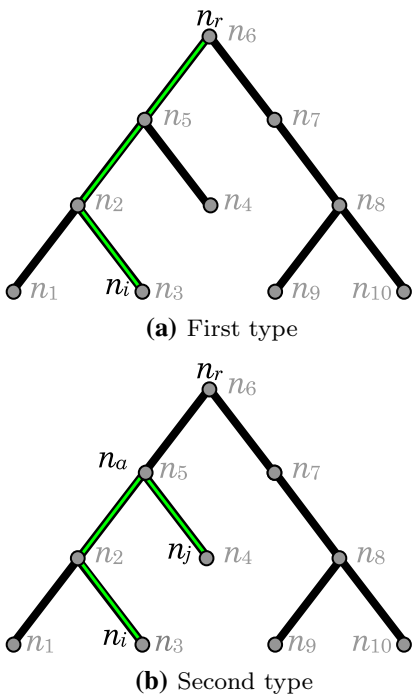
Appendix 1: Proof of Theorem 1

Theorem 1 *The proposed message-passing scheme considers all the possible paths of unrepeated nodes that the evader might travel in the RVG. Furthermore, it also considers any path of repeated nodes.*

Proof We first elaborate for the case when the RVG has a tree topology. Later, we built upon the tree case to address when the RVG has cycles within it.

The proof for the case of a RVG with tree topology proceeds as follows. First, it considers the paths of unrepeated nodes. This class of paths is exhaustively decomposed into two types, one that starts from the root node n_r to any other node $n_i \in RVG$ (refer to Fig. 19a), and the second, paths from any node $n_i \in RVG$ to any node $n_j \in RVG$, with $n_i \neq n_j \neq n_r$ (refer to Fig. 19b). Later, paths with repeated nodes are considered and it is shown that such class of paths can be constructed as concatenations of paths with unrepeated nodes.

First, we prove that the message-passing scheme considers the first type of unrepeated nodes paths. Let us consider then a path of unrepeated nodes from n_r to n_i, with $n_r \neq n_i$. When the message propagation starts from the leaf nodes all the way up to n_r, at some iteration it surely reaches node n_i, as n_r is the root node of the RVG and therefore n_i is a descendant node of n_r; subsequently, the message-passing scheme starts travelling the path from n_i up to n_r, which is the unrepeated nodes path from n_r to n_i but in reverse order, passing the necessary surveillance restrictions for going from n_r to n_i. The result follows.



(a) First type

(b) Second type

Fig. 19 Types of paths with unrepeated nodes

Now, we prove that the message-passing scheme considers the second type of unrepeated nodes paths. If the evader first visits n_r and then it decides to go to any other node $n_i \in RVG$, there is always the possibility that after the evader arrives to n_i, it might decide to move to any other node $n_j \neq n_r$, that is, to follow a path of the second type. So let us assume such case when the evader first visits n_r–using n_r as an entry point to the RVG–, then it arrives to a node n_i, and then it decides to follow a path of the second type to a node n_j. Focusing on these last two nodes, n_i and n_j, for each node a path can be tracked towards n_r until both paths have a common ancestor node, lets call it n_a, see Fig. 19b. Following this procedure the path with unrepeated nodes between n_i and n_j can be found (in a tree this path is unique). Such path can be divided into two parts, the first one from n_i to n_a and the second one from n_a to n_j. In the first part of the path, following the message-passing scheme we already know that node n_a receives the proper information or restrictions (from the parent of n_i, following a succession of ancestors nodes up to n_a), so that the pursuer can maintain SMV of an evader that first visits n_a and then follows a sequence of unrepeated nodes until n_i. If the pursuer fulfils the restrictions passed to n_a (indeed, this is done by first fulfilling the restrictions passed to n_r) then there must exist a path $\rho_p(n_a, n_i)$ from $gP(n_a)$ to $gP(n_i)$ for the pursuer, such that it allows evader surveillance when the evader moves from n_a to n_i; let us call that evader's path $\rho_e(n_a, n_i)$. If such path $\rho_p(n_a, n_i)$ exists, the pursuer can simply follow $\rho_p(n_a, n_i)$ in the opposite direction (that is $\rho_p(n_i, n_a)$, successfully returning to $gP(n_a)$) to maintain surveillance of an evader that follows $\rho_e(n_a, n_i)$, but also in the opposite direction (that is $\rho_e(n_i, n_a)$).[6] Furthermore, for every viable path for the pursuer from $gP(n_i)$ to $gP(n_a)$, there must be its counterpart in the opposite direction, that is from $gP(n_a)$ to $gP(n_i)$, because if that does not hold means that there exists a path $\rho_p(n_i, n_a)$ that the pursuer cannot follow in the opposite direction, which is not possible. Hence, no extra restrictions need to be passed to node n_a (and therefore to n_r) so that if the pursuer fulfils the restrictions passed to n_r, which includes the ones from n_a, the pursuer can maintain surveillance of an evader that moves starting from n_r, passing though n_a, then arriving to n_i and then moving back to n_a.

Let us analyse the second part of the path, that is when the evader moves from n_a to n_j. Following the message-passing

[6] Also recall that in a tree the path of unrepeated nodes from n_i to n_a is unique and it can only be $\rho_e(n_i, n_a)$.

scheme, n_a receives the proper information or restrictions, so that if the pursuer fulfils such restrictions it can maintain SMV of an evader that visits n_a and then follows a sequence of unrepeated nodes until it reaches any node below n_a, which also includes n_j; it can be seen that the message-passing scheme by itself considers the evader's path from n_a to n_j. Now, we can conclude that using the message passing scheme, node n_r receives all the necessary restrictions that the pursuer must fulfil to maintain surveillance of an evader that moves starting from n_r, passing though n_a, then arriving to n_i and then moving back to n_a and finally arriving to n_j.

Any subsequent path of repeated nodes of the evader can be described as concatenations of paths with unrepeated nodes. This is done splitting the repeated nodes paths by considering a new unrepeated nodes sub-path starting in a node before a repeated node appears. Therefore, every path of repeated nodes is a succession of paths of the type $n_i \rightarrow \cdots \rightarrow n_a \rightarrow \cdots \rightarrow n_j$, and first fulfilling the restrictions passed to n_r (using it as an entry point to the RVG), always allows the pursuer to fulfil the restrictions related to any node n_a, so a path $n_i \rightarrow \cdots \rightarrow n_a$ followed by $n_a \rightarrow \cdots \rightarrow n_j$ can always be traced by the pursuer as shown before, adding no extra restrictions to the ones already passed to n_r via the proposed message-passing scheme.

In the message passing-scheme, when the messages are passed from the leaf nodes all the way up to n_r, any node only receives the surveillance restrictions from all the nodes below it. By propagating the messages from n_r all the way down to the leaf nodes, the remaining surveillance restrictions are sent to every node $n_i \in RVG$ (restrictions from branches not bellow n_i), enabling any node n_i as an entry point to the RVG, i.e., as a root node. The result follows.

For the case of a RVG with cycles, it is possible to consider all the evader paths by unfolding the graph in a succession of trees (refer to Sect. 6.2 for a complete procedure to unfold the graph), therefore the result for this later case follows directly from the tree case. □

Appendix 2: Workspace partitions

The notion of SMV relies on the availability of a convex partition of the workspace W. In this section we present a two steps algorithm that generates two different resolution partitions, however, we first introduce some terminology used along this section.

Notice that reflex vertices are crucial in our problem, since they break the convexity of the polygonal environment.

Definition 5 For a given reflex vertex v, a *reflex ray* is a line segment from v, to the boundary of W, collinear to the supporting line of the edge incident to v.[7] Each reflex vertex defines two reflex rays. See Fig. 20a.

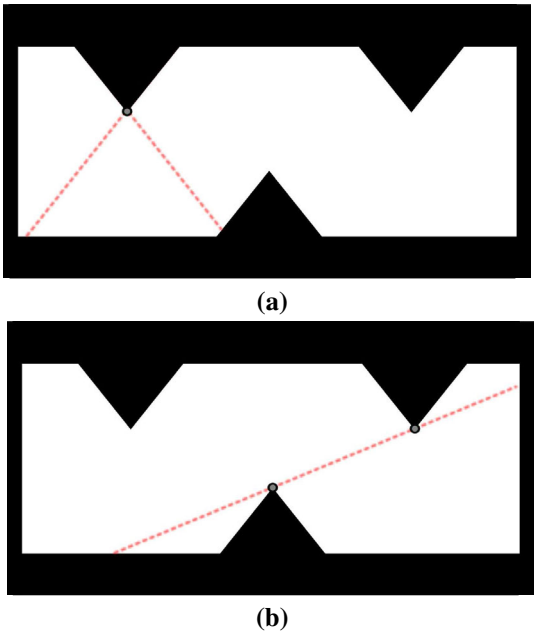


Fig. 20 **a** Reflex rays, and **b** extended bitangent segment

Definition 6 For a given pair of visible[8] reflex vertices, v and u, an *extended bitangent segment* $\mathcal{B}$ between them, is a line segment in the workspace W such that:

- The endpoints of $\mathcal{B}$ are neither v nor u.
- $\mathcal{B}$ is tangent to ∂W (boundary of W) in v and u.
- Considering a ball $B_r(x)$ with an arbitrarily small radius r centred at point x, then, $\forall x \in \mathcal{B}$, $B_r(x) \cap \partial W \cap \mathcal{B} = \{x\}$ or $B_r(x) \cap \partial W \cap \mathcal{B} = \emptyset$.
- There is no other extended bitangent segment $\mathcal{B}'$ related to v and u, such that $\mathcal{B} \subset \mathcal{B}'$.

Figure 20b shows an example of a extended bitangent segment.

Definition 7 A line segment $\overline{a\,b} \subset W$ is tangent to ∂W in a point p, iff:

- $p \neq a$ and $p \neq b$.
- Considering a ball $B_r(p)$ with an arbitrarily small radius r centred at p, then, $B_r(p) \cap \partial W \cap \overline{a\,b} = \{p\}$.

[7] Note that a reflex ray differs from an inflection ray, as defined in LaValle (2006). Whenever a robot crosses an inflection ray, a gap in the gap navigation tree (GNT) will (dis)appear.

[8] We say that two points are visible when we can draw a line segment between them and it is not intersected by any obstacle.

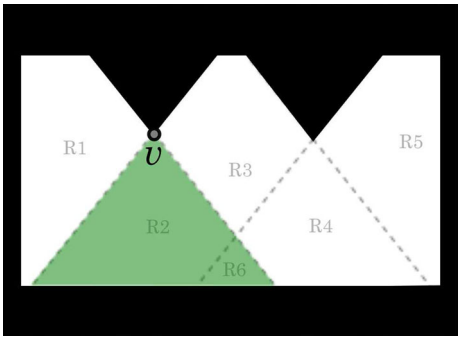


Fig. 21 $cgP(v) = \{R2, R6\}$, where both $R2$ and $R6$ are corner-guard regions

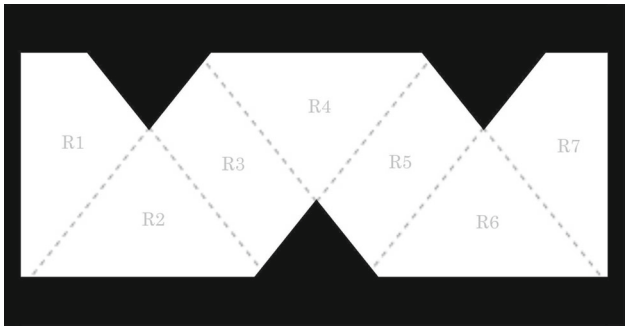


Fig. 22 Resulting partition after the first step of the partitioning algorithm

Definition 8 A *corner-guard polygon*, $cgP(v)$, for a given reflex vertex v, is the set of points within W delimited by the lines supporting v.[9] Refer to Fig. 21.

Our proposal for partitioning the environment follows a two-step approach, being *purely based on the environment structure rather than on the current positioning of the players*. The steps are as follows:

(1) For every reflex vertex, trace both of its reflex rays.
(2) For every pair of reflex vertices, trace an extended bitangent segment, whenever possible.

The first resolution partition comes from integrating reflex rays, considering a convex region R_i a convex polygon whose interior is not intersected by a reflex ray. Any convex region R_i is bounded by reflex rays, portions of them (as the reflex rays might intersect), or portions of the polygonal workspace boundary. The second resolution partition comes from considering both reflex rays and extended bitangent segments. In such case any convex region R_i might also be bounded by extended bitangent segments or portions of them.

First resolution partition (reflex rays)

Focusing on the first partition, notice that the first step of our approach yields for every reflex vertex, regions located over its related corner-guard polygons, which we name as corner-guard regions. Taking this into account, we get the next definition.

Definition 9 A region R_i is a *corner-guard region* for a given reflex vertex v, denoting it as $cgR(v)$, iff it is fully contained in $cgP(v)$, and at the same time, $\forall p \in cgR(v)$, $\overline{p\,v} \subset W$. Refer to Fig. 21.

Remark 7 A corner-guard polygon $cgP(v)$ might have more than one corner-guard region.

[9] *Presented* Murrieta-Cid et al. (2003)*, and referred in* Bhattacharya and Hutchinson (2011) *as star region.*

The relevance behind the concept of corner-guard regions is that, since the reflex vertices are the ones that break convexity in a given polygon, the evader is obliged to use them to break SMV with the pursuer, and once the pursuer is over a corner-guard region $cgR(v)$, the pursuer completely nullifies any attempt of the evader to break SMV with it making use *exclusively* of reflex vertex v. This happens because there is no point $p \in cgR(v)$, where an open line segment is traceable to a point $q \in R_i$, such that $\overline{p\,q}$ is *exclusively* intersected by the edges incident to v. Our fist resolution partition generates the corner-guard regions, which nullifies all possible evader escape using only one single reflex vertex. An example of the resulting partition after applying the first step of the partitioning algorithm is shown in Fig. 22.

How to refine a given convex partition (pivot segments)

Intuitively, since maintaining SMV is a sufficient condition for surveillance, it is desirable to choose a partition that yields pairs of SMV regions that are as large as possible, and yields regions that are SMV with as many other regions as possible. Notice that both objectives have a trade-off between them as we can keep splitting any region. On the other hand, we are also interested on keeping reflex rays as they generate corner-guard regions, nevertheless, we can still do a further refinement to the partition yielded from the first step of our approach, in order to generate new regions that are as large as possible and have a gain in SMV with respect to its regions of provenance. Keeping that in mind, denote by $SMV(R_i)$ the set of regions that are SMV with region R_i. The total area of these regions is then given by

$$\mathcal{V}_{\text{sm}}(R_i) = \sum_{R_k \in SMV(R_i)} \mu(R_k)$$

in which $\mu(R_k)$ denotes the area of region R_k. If the summation of each $\mathcal{V}_{\text{sm}}(R_i)$ is done over all region R_i, then $\sum \mathcal{V}_{\text{sm}}(R_i)$ is a global measure intrinsic to a given partition, which depicts us a degree of interaction between its regions

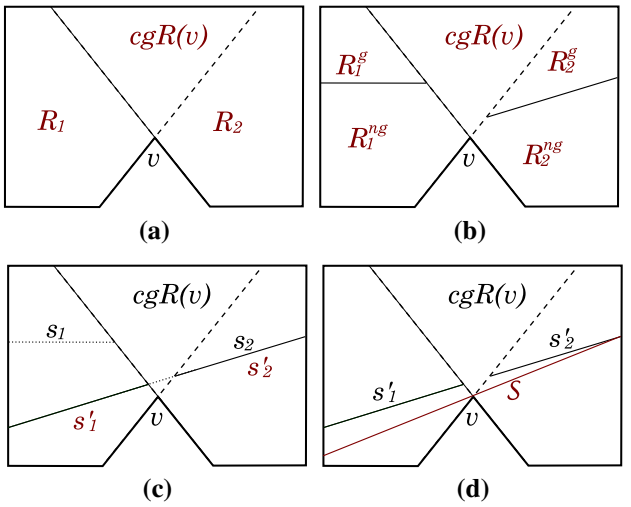


Fig. 23 Different partitioning procedures depicted on Theorem's 5 proof

regarding to SMV. As it comes naturally from looking for a partition where every region is of the largest size and with the largest SMV relation, we are interested on a partition that gives a value of $\sum \mathcal{V}_{\text{sm}}(R_i)$ as big as possible.

After applying the first step of our partitioning algorithm, we can then perform a further refinement of the yielded partition in order to increment $\sum \mathcal{V}_{\text{sm}}(R_i)$. Knowing that the reflex vertices are the features that break the convexity of the environment, without loss of generality, let us focus on a single reflex vertex v and assume that it only has a single corner-guard region related to it, further referred as $cgR(v)$, and call R_1 and R_2, to the two regions adjacent to $cgR(v)$ (see Fig. 23a). We know that $cgR(v)$ is SMV with both R_1 and R_2, but R_1 and R_2 are not SMV with each other. To bring an increment on $\sum \mathcal{V}_{\text{sm}}(R_i)$, what we need to do is to partition both R_1 and R_2 in order to generate new regions over R_1 and R_2 that are now SMV. There are several ways for partitioning R_1 and R_2, however, in Theorem 5 we show that the best way to partition R_1 and R_2 in order to bring an increment to $\sum \mathcal{V}_{\text{sm}}(R_i)$, is to add pivot segments over v, consequently, let us first define a pivot segment.

Definition 10 A *pivot segment* $\mathcal{S}$ for a given reflex vertex v, is a line segment in W such that:

- The endpoints of $\mathcal{S}$ are not v.
- $v \in \mathcal{S}$.
- Considering a ball $B_r(x)$ with an arbitrarily small radius r centered at point x. $\forall x \in \mathcal{S}$, $B_r(x) \cap \partial W \cap \mathcal{S} = \{x\}$ or $B_r(x) \cap \partial W \cap \mathcal{S} = \emptyset$.
- There is no other pivot segment $\mathcal{S}'$ related to v, such that $\mathcal{S} \subset \mathcal{S}'$.

Theorem 5 *For a given reflex vertex v, for any other procedure ς for partitioning R_1 and R_2 into two new regions each, apart from drawing pivot segments, there exists a pivot segment $\mathcal{S}$ that partitions both R_1 and R_2 that yields a bigger value of $\sum \mathcal{V}_{\text{sm}}(R_i)$ than the one related to the partition obtained by applying ς.*

Proof First of all, we know that vertex v stands between regions R_1 and R_2, so they are not SMV. In order to bring an increment to $\sum \mathcal{V}_{\text{sm}}(R_i)$ in the simplest way, R_1 and R_2 should be split into two new regions each, such that some of the new emerging regions are now SMV. Let us call the new regions, R_1^g, R_1^{ng}, R_2^g and R_2^{ng}, where g stands for the fact that region R_i^g has a gain in visibility because it is now SMV with a region at the other side of $cgR(v)$, and ng stands for the case where there is no gain in SMV for a region R_i^{ng} (Fig. 23a, b).

What it is done next, is to characterize the procedure ς for partitioning the regions R_1 and R_2. To further partition R_1 and R_2 into two new regions each, also recalling that the resulting regions must be convex, both R_1 and R_2 are split with a line segment each. Therefore, the procedure ς, in order to be valid, must take the form of adding a couple of line segments that respectively split R_1 and R_2. Also, for ς to be valid, regions R_1^g and R_2^g should emerge because if there is no gain in SMV, then there is no sense on applying ς.

Now let us assume that ς is then a valid partitioning procedure, and that it adds two line segments s_1 and s_2, one for each respective region R_1 and R_2, but such that the endpoints of both s_1 and s_2 are not collinear (see Fig. 23b). If this is the case, there is always a procedure ς' that adds two line segments s_1' and s_2', being all its four endpoints collinear, such that gives a greater value of $\sum \mathcal{V}_{\text{sm}}(R_i)$. To obtain such a procedure ς' from ς, we only need to extend either s_1 or s_2 towards ∂W, subsequently keep the segment s_j whose extension permits an increment on size for R_i^g, and then make $s_j' = s_j$ and s_i' equal to the intersection of the extension of s_j with R_i (see Fig. 23c). Applying this transformation is equivalently to growing region R_i^g towards R_i^{ng}, such that R_i^g is still SMV with R_j^g.

Let us consider a region R_k within a given partition, and call R_{nb} to a neighbor region of R_k, both sharing a boundary bigger than a single point. Then, define the next sets: $\mathcal{I} = SMV(R_k) \cap SMV(R_{nb})$, $\mathcal{E}_k = SMV(R_k) \setminus \mathcal{I}$ and $\mathcal{E}_{nb} = SMV(R_{nb}) \setminus \mathcal{I}$. The set $\mathcal{I}$ contains the regions SMV with both R_k and R_{nb}, the set $\mathcal{E}_k$ contains the regions exclusively SMV with region R_k, and the set $\mathcal{E}_{nb}$ contains the regions exclusively SMV with region R_{nb}.

Under such definitions and returning to the resulting partition from applying ς, see Fig. 23b, make $R_k = R_1^g$ and $R_{nb} = R_1^{ng}$, then we have that $\mathcal{I} = \{cgP(v), R_1^g, R_1^{ng}\}$,

$\mathcal{E}_k = \{R_2^g\}$ and $\mathcal{E}_{nb} = \emptyset$, therefore growing R_k towards R_{nb} keeps $\mathcal{E}_{nb} = \emptyset$, the terms $\mathcal{V}_{\text{sm}}(R_i)$ for any $R_i \in \mathcal{I}$ remains equal, but terms $\mathcal{V}_{\text{sm}}(R_i)$ for any $R_i \in \mathcal{E}_k$ have an increment if R_k grows towards R_{nb} up to s_1', as $\mu(R_k)$ increases, producing an increment to $\sum \mathcal{V}_{\text{sm}}(R_i)$. We conclude that procedure ς' yields a bigger value of $\sum \mathcal{V}_{\text{sm}}(R_i)$ than the one obtained from ς, then, our current best way to partition R_1 and R_2 is to add a couple of line segments s_1 and s_2 whose endpoints are collinear.

Now let us consider again procedure ς', which we assume to be valid and consists on adding a couple of segments s_1' and s_2', whose endpoints are collinear, but now add the restriction that the straight line that contains both s_1' and s_2' does not contain v. If this is the case, we can always generate a pivot segment $\mathcal{S}$ from ς', which yields a greater value of $\sum \mathcal{V}_{\text{sm}}(R_i)$ than the one yielded from ς', all by just following the next steps: choose either s_1' or s_2', and name it as s_i', then trace a line segment from the endpoint of s_i' in contact with ∂W, towards v location, extending it as long as possible (see Fig. 23d). Let us call such procedure $\varsigma^\star$. $\mathcal{S}$ now defines new potential boundaries for R_1^g and R_2^g. This procedure would be equivalent to both growing R_1^g towards R_1^{ng}, and R_2^g towards R_2^{ng}, growing both regions as much as possible still maintaining R_1^g and R_2^g mutually visible (see Fig. 23d). Following equivalent arguments to the ones used to compare ς against ς', it is shown that the partition yielded by $\varsigma^\star$ gives a greater value of $\sum \mathcal{V}_{\text{sm}}(R_i)$ than the one obtained through ς'. Since our best current method to partition R_1 and R_2 was to add a couple of segments s_1 and s_2 whose endpoints are collinear, and since we have proved that within that category the best option is to trace a pivot segment $\mathcal{S}$, then the result follows. □

It is interesting to notice that as we increase the number of pivot segments that partition the environment touching a reflex vertex, the number of regions that are SMV also increases, and $\sum \mathcal{V}_{\text{sm}}(R_i)$ also increases, see Fig. 24. In the same sense, Theorem 5 is of importance because it tells us that no matter how many lines we want to add to refine the current environment's partition in order to increment $\sum \mathcal{V}_{\text{sm}}(R_i)$, the lines that we should add should be pivot segments. Indeed, if the number of partitioning pivot segments tends to infinity, SMV tends to classical visibility, therefore, a solution based on SMV is a solution for classical visibility for a given resolution. It is possible to think about this as if we had a line pivoting over the reflex vertex, starting being parallel to one of the reflex rays, and ending up being parallel to the other reflex ray. However, the behavior of the algorithms to compute the safe-areas, and the approximation of the solution for a given partition to classical visibility, do depend on the used partition, and a deeper analysis is left for future work.

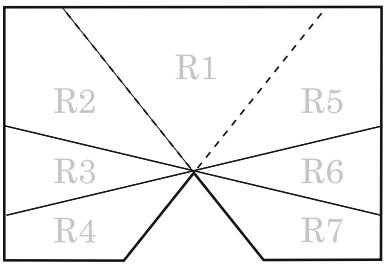


Fig. 24 Pivot segments

Remark 8 Alternatively, Theorem 5 states that no matter how many lines we want to add to refine a current environment's partition in order to increase $\sum \mathcal{V}_{\text{sm}}(R_i)$, the lines that we should add must be pivot segments. Furthermore, if the number of partitioning pivot segments tends to infinity, SMV tends to classical visibility, therefore, a solution based on SMV is a solution for classical visibility for a given resolution.

Second resolution partition (reflex rays and extended bitangent segments)

Actually, what we propose in our second step of our partitioning algorithm is to add pivot segments between reflex rays in order to bring a further refinement to the partition yielded from the first step. The pivot segments that we add are the extended bitangent segments, which happen to be pivot segments that a pair of reflex vertices have in common. Adding them results into a further refinement that increases $\sum \mathcal{V}_{\text{sm}}(R_i)$, which seeks to avoid generating a big number of regions that will lately translate into more computational work for the surveillance algorithms of the previous sections.

The complete two steps partitioning procedure is based on the notion of visibility complex (Pocchiola and Vegter 1996). However, to construct our partition we make use of some specific maximal free segments (reflex rays and extended bitangent segments) in order to yield equivalence classes (the regions $\{R_i\}$), where the positions within a class (R_i) share the same approximation to the visibility polygon [the approximation is $SMV(R_i)$]. An example of the resulting partition after applying the second step of the partitioning algorithm is shown in Fig. 25.

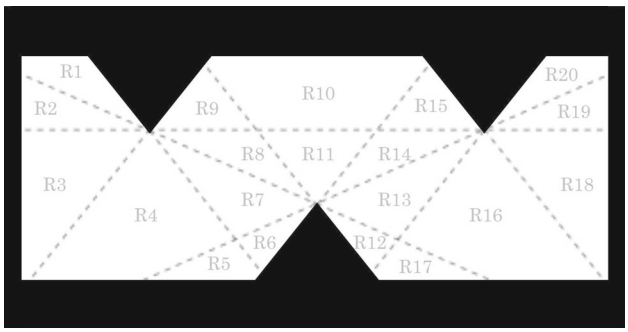


Fig. 25 Second resulting partition

General notation	
$\{R_i\}$ or $R_1, \ldots, R_n$	Convex environment partition
R_i or R	Convex region
W	Workspace
∂W	Workspace boundary
MVG	Mutual visibility graph
$SMV(R)$	Set of regions that are SMV with region R
AG	Accessibility graph
RVG	Reduced visibility graph
n_i	Node in RVG
E	Current evader's region
P	Current pursuer's region
V_e	Evader's speed
V_p	Pursuer's speed
q_i	Critical point
$q_1 \rightarrow q_2 \rightarrow \cdots \rightarrow q_{n-1} \rightarrow q_n$	Critical points sequence
$gP(q)$	Guard polygon of point q
t_e^k	Smallest time that takes to the evader to reach the critical point $k+1$ from its k Critical point
t_p^k	Smallest time that takes the pursuer to reach its corresponding $k+1$ guard polygon departing from the k Guard polygon
$sA(n_i)$	Safe area
$q_1 \rightarrow q_2 \rightarrow \cdots \rightarrow q_{n-1} \rightarrow q_1$	Critical points cycle
Algorithm and pursuit strategies specific notation	
n_r	Root node in RVG
C_{ij}^k	The kth cluster of regions from n_i towards n_j
o_{ij}^k	The transition point between clusters C_{ij}^k and C_{ij}^{k+1}
l_{ij}^k	The kth sub-edge over line segment that joins nodes n_i and n_j
$SMV(cR_{ij}^k)$	Set of regions SMV with cluster cR_{ij}^k
$sA(q_1)^k$	Safe area calculated in the kth iteration of the cycles algorithm
S	Set of local solution
$\mathbf{V}$	Set of reflex vertices used to calculate the S set
$t_e(e, v)$	Smallest time that takes to the evader to travel from its current position e to point v
$t_p(q, sA(v))$	Smallest time that takes to the pursuer to reach $sA(v)$ from point q, respecting surveillance restrictions between $sA(v)$ and q
e_{init}	Initial evader's position
Decidability specific notation	
$\mathbf{R}_{i,i+1}$	Relation that models a mapping between $gP(q_i)$ and $gP(q_{i+1})$
$Im^{-1}(\mathbf{R}_{i,i+1})$	Preimage of $R_{i,i+1}$
$\mathbf{R}_{i+1,i+2} \circ \mathbf{R}_{i,i+1}$	Composition of relations $\mathbf{R}_{i,i+1}$ and $\mathbf{R}_{i+1,i+2}$
$\mathbf{R}_{1,n}$	Composition of relations $\mathbf{R}_{1,2}$, $\mathbf{R}_{2,3}$, ..., $\mathbf{R}_{n-1,n}$
$\mathbf{R}_{1,n}^N$	Relation of $N-1$ compositions of relation $\mathbf{R}_{1,n}$
$\mathbf{R}_{1,n}^\infty$	Relation of an infinite number of compositions of relation $\mathbf{R}_{1,n}$
Complexity specific notation	
(V, A)	$RVG = (V, A)$; V is set of vertices; A is set of edges
V_{ratio}	V_p/V_e
Λ	Size of the paths that the evader travels in terms of the number of critical points that it visits
Π	Set of particular properties that define the evader's paths
Partition specific notation	
$\mathcal{B}$	Bitangent segment
v or u	Reflex vertices
$cgP(v)$	The corner-guard polygon related to reflex vertex v
$cgR(v)$	A corner-guard region related to reflex vertex v
$\mu(R_i)$	Area of R_i
$\mathcal{V}_{\text{sm}}(R_i)$	Total area of regions SMV with R_i
$\mathcal{S}$	Pivot segment
ς	A procedure to refine an environment's partition

References

Bandyopadhyay, T., Li, Y., Ang, M.H., & Hsu, D. (2006). A greedy strategy for tracking a locally predictable target among obstacles. In *Proceedings of IEEE International Conference on Robotics and Automation*.

Bandyopadhyay, T., Ang, M.H., & Hsu, D. (2007). Motion planning for 3-D target tracking among obstacles. In *International Symposium on Robotics Research*.

Barrière, L., Flocchini, P., Fraigniaud, P., & Santoro, N. (2002). Capture of an intruder by mobile agents. In *Proceedings of the 14th Annual ACM Symposium on Parallel Algorithms and Architectures* (pp. 200–209). Winnipeg, Manitoba.

Başar, T., & Olsder, G. (1982). *Dynamic noncooperative game theory*. London: Academic Press.

Becerra, I., Murrieta-Cid, R., & Monroy, R. (2010). Evader surveillance under incomplete information. In *IEEE International Conference on Robotics and Automation*.

Becker, C., González-Baños, H., Latombe, J.-C., & Tomasi, C. (1995). An intelligent observer. In *International Symposium on Experimental Robotics*.

Bhattacharya, S., & Hutchinson, S. (2009). On the existence of nash equilibrium for a two player pursuit-evasion game with visibility constraints. In *International Journal on Robotics Research*.

Bhattacharya, S., & Hutchinson, S. (2011). A Cell decomposition approach to visibility-based pursuit evasion among obstacles. *The International Journal of Robotics Research*, *30*(14), 1709–1727.

Chung, T.H. (2008). On probabilistic search decisions under searcher motion constraints. In *WAFR 2008* (pp. 501–516).

Chung, T., Hollinger, G., & Isler, V. (2011). Search and pursuit-evasion in mobile robotics: A survey. *Autonomous Robots*, *31*(4), 299–316.

Cormen, T. H., Stein, C., Rivest, R. L., & Leiserson, C. E. (2001). *Introduction to algorithms*. Groveport: McGraw-Hill Higher Education.

Efrat, A., Gonzalez-Baños, H.H., Kobourov, S.G., & Palaniappan, L. (2003). Optimal motion strategies to track and capture a predictable target. In *Proceedings of IEEE International Conference on Robotics and Automation* (pp. 411–423). Taipei, Taiwan.

Fabiani, P., & Latombe, J.-C. (1999). Tracking a partially predictable object with uncertainty and visibility constraints: A game-theoretic approach. In *IJCAI*.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability*. New York: W. H. Freeman and Company.

González, H.H. et al. (2002). Real-time combinatorial tracking of a target moving unpredictably among obstacles. In *Proceedings of IEEE International Conference on Robotics and Automation*.

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

Hájek, O. (1965). *Pursuit games*. New York: Academic Press.

Hespanha, J., Prandini, M., & Sastry, S. (2000). Probabilistic pursuit-evasion games: A one-step Nash approach. In *Proceedings of Conference on Decision and Control*.

Hollinger, G., Singh, S., Djugash, J., & Kehagias, A. (2009). Efficient multi-robot search for a moving target. *The International Journal of Robotics Research*, *28*(2), 201–219.

Isaacs, R. (1965). *Differential games*. New York: Wiley.

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

Jung, B., & Sukhatme, G. (2002). Tracking targets using multiple robots: The effect of environment occlusion. *Journal Autonomous Robots*, *12*, 191–205.

Latombe, J.-C. (1991). *Robot motion planning*. New York: Kluwer Academic Publishers.

LaValle, S.M., González-Baños, H.H., Becker, C., & Latombe, J.-C. (1997) Motion strategies for maintaining visibility of a moving target. In *Proceedings of IEEE International Conference on Robotics and Automation*.

LaValle, S. M. (2006). *Planning algorithms*. Cambridge: Cambridge University Press.

Merz, A.W. (1971). The homicidal chauffeur a differential game. *PhD. Thesis*. Stanford University.

Murrieta-Cid, R., Sarmiento, A., & Hutchinson, S. (2003). On the existence of a strategy to maintain a moving target within the sensing range of an observer reacting with delay. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Murrieta-Cid, R., Tovar, B., & Hutchinson, S. (2005). A sampling-based motion planning approach to maintain visibility of unpredictable targets. *Journal Autonomous Robots*, *19*(3), 285–300.

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

Murrieta-Cid, R., Monroy, R., Hutchinson, S., & Laumond, J.-P. (2008). A complexity result for the pursuit-evasion game of maintaining visibility of a moving evader. In *IEEE International Conference on Robotics and Automation*.

O'Rourke, J. (2000). *Computational geometry in C*. Cambridge: Cambridge University Press.

O'Rourke, J. (1987). *Art gallery theorems and algorithms*. Oxford: Oxford University Press.

O'Kane, J.M. (2008). On the value of ignorance: Balancing tracking and privacy using a two-bit sensor. In *Proceedings of International Workshop on the Algorithmic Foundations of Robotics*.

Parker, L. (2002). Algorithms for multi-robot observation of multiple targets. *Journal Autonomous Robots*, *12*, 231–255.

Parsons, T. D. (1976). Pursuit-evasion in a graph. In Y. Alani & D. R. Lick (Eds.), *Theory and application of graphs* (pp. 426–441). Berlin: Springer.

Pocchiola, M., & Vegter, G. (1996). The visibility complex. In *International Journal of Computational Geometry and Applications*.

Rappoport, A. (1966). *Two-person game theory*. Dover: Courier Corporation.

Sachs, S., Rajko, S., & LaValle, S. M. (2004). Visibility-based pursuit-evasion in an unknown planar environment. *International Journal on Robotics Research*, *23*(1), 3–26.

Shermer, T.C. (1992). Recent results in art galleries. In *Proceedings of the IEEE* (vol. 80, no. 9).

Stump, E., Michael, N., Kumar, V., & Isler, V. (2011). Visibility-based deployment of robot formations for communication maintenance. In *IEEE International Conference on Robotics and Automation 2011*.

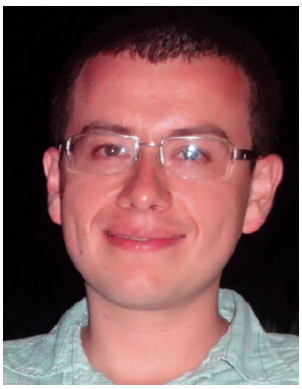
Suzuki, I., & Yamashita, M. (1992). Searching for a mobile intruder in a polygonal region. *SIAM Journal on Computing*, *21*(5), 863–888.

Takeuti, G., & Zaring, W. M. (1971). *Introduction to axiomatic set theory*. New York: Springer.

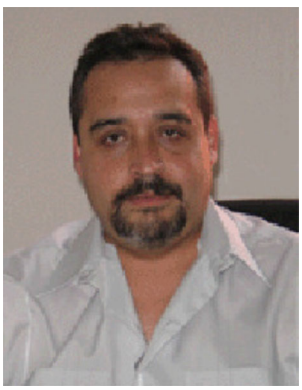
Tekdas, O., Yang, W., & Isler, V. (2010). Robotic routers: Algorithms and implementation. *The International Journal of Robotics Research*, *29*(1), 110–126.

Tovar, B., & LaValle, S. M. (2008). Visibility-based pursuit—evasion with bounded speed. *The International Journal of Robotics Research*, *27*(11–12), 1350–1360.

Vidal, R., et al. (2002). Probabilistic pursuit-evasion games: Theory, implementation, and experimental evaluation. *IEEE Transactions on Robotics and Automation*, *18*(5), 662–669.



Israel Becerra received the B.S. degree in Mechatronics Engineering from the ITESM, campus Estado de México (2007), and the M.Sc. degree in Computer Science from CIMAT—Centro de Investigación en Matemáticas—(2010). Currently (2013) he is pursuing the Ph.D. degree in Computer Science at CIMAT, Guanajuato, Mexico. He is mainly interested in motion planning, pursuit-evasion games and probabilistic robotics.



Rafael Murrieta-Cid received the B.S. degree in Physics Engineering from the ITESM, campus Monterrey (1990). He received his Ph.D. from the Institut National Polytechnique (INP) of Toulouse, France (1998). His Ph.D. research was done in the RIA group of the LAAS-CNRS. In 1998–1999, he was a postdoctoral researcher in the Computer Science Department at Stanford University. In 2002–2004, he was working as a postdoctoral research associate in the Beckman Institute and Department of Electrical and Computer Engineering of the University of Illinois at Urbana-Champaign. From August 2004 to January 2006 he was a Professor and Director of the Mechatronics Research Center in Tec de Monterrey, Campus Estado de México.

Since March 2006, he has been working in the Mathematical Computing Group at the CIMAT —Centro de Investigación en Matemáticas, in Guanajuato México. He is mainly interested in robotics and robot motion planning, and has published more than 40 papers in Journals and International Conferences on these topics.

Raul Monroy obtained a Ph.D. in Artificial Intelligence in 1998 from Edinburgh University, under the supervision of Professor Bundy. He has been in Computing at Tecnológico de Monterrey (ITESM), Campus Estado de México, since 1985. In 1992 he was promoted to Assistant Professor and in 2000 he was promoted to Associate Professor. Since 1998 he is a member of the CONACYT-SNI National Research System. Dr. Monroy has held 6 research Grants from several funding agencies, including CONACYT (holder), the national research council, BMBF (co-holder), FRIDA (holder) and CONACyT-REDII (co-holder). He is the sole or joint author of over 20 published papers. He is programme co-chair for MICAI-2004 and MICAI-2005 and has been on several programme committees. He has been Secretary Treasurer to the Mexican Society for Artificial Intelligence since 2000. Dr. Monroy's research focuses on automating the use of theorem proving to formal methods of system development. He is also interested in issues of computer security. Currently, his research concerns: the discovery and application of proof plans to automate the verification of security protocols; the discovery an application of general search control strategies for uncovering and correcting errors in either a system or its specification; and the discovery of novel methods for anomaly detection in computer security.

Seth Hutchinson received his Ph.D. from Purdue University in 1988. In 1990 he joined the faculty at the University of Illinois in Urbana-Champaign, where he is currently a Professor in the Department of Electrical and Computer Engineering, the Coordinated Science Laboratory, and the Beckman Institute for Advanced Science and Technology. Hutchinson has served as Editor-in-Chief of the IEEE Transactions on Robotics, and as the first Editor-in-Chief for the RAS Conference Editorial Board. He currently serves on the editorial boards of the International Journal of Robotics Research and the Journal of Intelligent Service Robotics. In 1996 he was a Guest Editor for a special section of the Transactions devoted to the topic of visual servo control, and in 1994 he was co-chair of an IEEE Workshop on Visual Servoing. In 1996 and 1998 he co-authored papers that were finalists for the King-Sun Fu Memorial Best Transactions Paper Award. He was co-chair of IEEE Robotics and Automation Society Technical Committee on Computer and Robot Vision from 1992 to 1996, and has served on the program committees for more than fifty conferences related to robotics and computer vision. He has published more than 150 papers on the topics of robotics and computer vision, and is coauthor of the books "Principles of Robot Motion: Theory, Algorithms, and Implementations," published by MIT Press, and "Robot Modeling and Control," published by Wiley. Hutchinson is a Fellow of the IEEE.

Jean-Paul Laumond received the M.S. degree in Mathematics, the Ph.D. degree in Robotics and the Habilitation degree from the University Paul Sabatier, Toulouse France, in 1976, 1984 and 1989 respectively. He is Directeur de Recherche at LAAS-CNRS in Toulouse, France. With his group Gepetto (www.laas.fr/gepetto), he is exploring the computational foundations of anthropomorphic motion. He has been coordinator of two the European Esprit projects PROMotion (Planning RObot Motion, 1992–1995) and MOLOG (Motion for Logistics, 1999–2002), both dedicated to robot motion planning technology. During 2001 and 2002 he created and managed Kineo CAM, a spin-off company from LAAS-CNRS devoted to develop and market motion planning technology. His current research is devoted to Human Motion studies along three perspectives: artificial motion for humanoid robots, virtual motion for digital actors and mannequins, and natural motions of human beings. He teaches Robotics at ENSTA and Ecole Normale Supérieure in Paris. He has edited three books. He has published more than 150 papers in International Journals and Conferences in Computer Science, Automatic Control and Robotics. He is IEEE Fellow and Member of the IEEE RAS AdCom.