Noname manuscript No.
(will be inserted by the editor)

A Visibility-based Pursuit-Evasion Game between Two Nonholonomic Robots in Environments with Obstacles

Eliezer Lozano · Israel Becerra · Ubaldo Ruiz · Luis Bravo · Rafael Murrieta-Cid

the date of receipt and acceptance should be inserted later

Abstract In this paper, a visibility-based pursuit-evasion game in an environment with obstacles is addressed. A pursuer wants to maintain the visibility of an evader at all times. Both players are nonholonomic robots shaped like discs. To determine the players' motion policies and their trajectories–subject to differential constraints–, an RRT* approach that minimizes the time traveled is utilized. The proposed formulation presents an alternative for computing a strategy of persistent surveillance of the evader, difficult to model from a classical differential games perspective given that there is no clear termination condition when the pursuer can maintain the evader's visibility forever. A sufficient condition to keep evader surveillance is also provided. Additionally, the proposed approach is general because it can be adapted to address a variety of scenarios. To illustrate such flexibility, we address different aspects of the problem: 1) Knowledge of the environment (availability of a global map vs. a local representation). 2) Strategies of the players (execution of optimal strategies vs. deviations from the optimal ones to deceive the opponent). 3) Sensor capabilities (limited vs. unlimited sensor range).

Keywords Pursuit-evasion · Motion planning · Nonholonomic constraints · Incomplete information · Algorithms

This work was financially supported by CONACYT Alliance of Artificial Intelligence, Catedras CONACyT projects 745 and 1850, CONACYT grant A1-S-21934 and Intel Corporation.

Eliezer Lozano, Israel Becerra, Luis Bravo, Rafael Murrieta-Cid
Centro de Investigación en Matemáticas (CIMAT), Guanajuato, Mexico
E-mail: {eliezer.lozano,israelb,luixbravo,murrieta}@cimat.mx

Ubaldo Ruiz
Centro de Investigación Científica y de Educación Superior de Ensenada (CICESE), Baja California, Mexico
Corresponding author
E-mail: uruiz@cicese.mx

Israel Becerra, Ubaldo Ruiz
are also with Consejo Nacional de Ciencia y Tecnología (CONACYT), Mexico City, Mexico

1 Introduction

In this paper, we address a visibility based pursuit-evasion game [23,6,3]. The pursuer's goal is to maintain the evader's visibility at all times while the evader wants to avoid the pursuer's surveillance. In our modeling, both players are differential drive robots (DDR), shaped like discs with the same radius. A DDR is a nonholonomic system [22]; it has two wheels with two independent motors, which allow the robot to rotate in place. It is important to note that the DDR cannot move instantaneously in all directions. That is the intuition behind a nonholonomic system. A formal treatment of the nonholonomic property can be found in [22]. The game takes place in a 2D environment with obstacles; the obstacles generate both motion and visibility constraints. The pursuer sees the evader if a clear line of sight connecting the center of the circular robots does not intersect an obstacle.

A diversity of approaches have been proposed to address the visibility-based pursuit-evasion game. For instance, in [39], the authors proposed an analytical solution and a dominant strategy for a pursuer to maintain an evader's visibility, considering a single corner. However, conversely to our modeling, in that work, both players are considered as holonomic points. Moreover, in the present paper, several corners are considered simultaneously, generating multiple potential escape regions. Also, in [37], the authors addressed the problem

of moving a camera to maximize the visibility of a group of targets. Their proposed algorithm is based on visibility integrity, which is used to build the so-called visibility integrity roadmap (VIR) queried to plan a surveillance strategy. Nonetheless, no guarantees of surveillance are provided.

Alternative pursuit-evasion games have been previously proposed. In one of them, the pursuer wants to capture the evader [15,30]. The classical Isaacs methodology [15] is a commonly used tool to address such a game. It starts assuming that capture is achieved, and then the players' trajectories are retrieved backward in time. Note that the Isaacs methodology does not provide a way to compute the evader's trajectories or strategies when the evader escapes, that is, when the evader prevents capture forever. The visibility-based pursuit-evasion game addressed in this paper is dual to that capture game, in the sense that the pursuer wins when it can maintain the evader's visibility forever. Indeed, it is unclear how to model the game from a termination condition since the pursuer's objective is to keep the evader within its visibility region indefinitely. There are practical examples where the addressed tracking problem is relevant: a robotic system that allows the surveillance of a malicious agent that can damage human or remote monitoring of industrial processes using autonomous observers capable of maintaining surveillance of an event. Indeed, the latter is an example where the monitoring task needs to be endless in practice. Consequently, we model the game using a methodology different from the one described by Isaacs. There are two relevant times in the proposed approach: 1) The time needed for the evader to reach the pursuer's visibility region's border. 2) The time required by the pursuer to reach a region in the environment that prevents the evader from escaping. Those two times are compared to compute the players' motion strategies and establish a sufficient condition to keep evader surveillance.

The evader moves from its position to the border of the pursuer's visibility region following the minimum time trajectory to escape. In contrast, the pursuer travels the minimum time trajectories from its current position to critical regions that prevent the evader's escape. To find the shortest time trajectories for the nonholonomic DDRs in an environment with obstacles, we use an RRT* algorithm [19,18] in which the local planner employed to connect two nodes is based on the method proposed in [2]. The RRT* is asymptotically optimal; however, we can expect that the method yields right enough approximations to the optimal players' trajectories for a sufficiently large number of nodes.

In the proposed formulation, there is a one-to-one correspondence between the evader's escape regions and the pursuer's prevention from escape regions, resulting from their intrinsic relationship with the environment's vertices. This finding crucially helps to model the problem; however, there is an extra complication. The pursuer's visibility region changes as the pursuer moves; consequently, its borders corresponding to the evader's goals also change. To model that evader's dynamic goal, similarly to [17], the pursuer's trajectory to reach a prevention region is computed first. After that, a group of planning problems must be solved for the evader, in which the evader's trajectories to reach a set of goals corresponding to the dynamic escape region are obtained. The computed trajectories are used to decide the players' strategies; more precisely, a worst-case for the pursuer is identified as the pair of prevention and escape regions (hence, environment vertex) that maximizes the difference of duration between the respective shortest trajectories of the players.

It is worth mentioning that the proposed approach performs short-term plans. The issue with several steps ahead strategies is that they further increase the processing time required to compute them. Indeed, it can be proved that the tracking problem in a polygonal environment is NP-Complete [3]. We call short-term plans the procedure that chooses the goal regions for the players taking into account only a single visit in the future to a pair of regions, in contrast, to compute a sequence of visits to several pairs. Nonetheless, our approach can be implemented in a closed-loop fashion. The pair of pursuer's and evader's trajectories *are executed* for the pursuer's worst-case, but the players only assume the respective goals for a short amount of time. After that amount of time, the players' best goals might have changed; moreover, the evader had the freedom to choose not to move along its best shortest time trajectory. Thus, another iteration of the surveillance algorithm is performed to keep the plan updated. In the following, an evader (pursuer) that sticks to the worst-case strategy will be referred to as a rational evader (pursuer); otherwise, it will be referred to as an irrational one. In other words, an irrational player can choose a sub-optimal goal or action in an attempt to deceive the other player. An example of this might be a moving person whose future motion is random, thus tough to predict. Such a scenario might correspond to an autonomous robot that carries a person's items as she moves in the environment. Our formulation does not imply any cooperation between the players. In the present work, an analysis is provided on the time window at which the pursuer needs to correct its strategy to keep surveillance of an irrational evader.

Finally, our formulation is flexible enough to be adjusted to consider different aspects of the problem. In

particular, we address the following three aspects: 1) Knowledge of the environment. Variants are considered where the players know the global map and when only a local map is available. 2) Types of player's strategies. The approach is designed to be able to deal with rational and irrational players. 3) Sensor capabilities. We explore the cases where the sensor has a limited or unlimited range.

1.1 Previous work

In general, in pursuit-evasion, a robot or a team of robots wants to chase other robots. A pursuit-evasion problem can be defined in several ways. A variant considers several pursuers that have the task of finding an evader in an environment with obstacles [13,16,11, 34]. A recent survey of pursuit-evasion problems is presented in [9]. Another variant consists of maintaining a moving evader's visibility in an environment populated with obstacles [3,27,8,7]. The third variant of a pursuit-evasion problem is to give the pursuer the task of capturing a moving evader [15,30,1,10,26], that is, moving to a contact configuration or closer than a given capture distance.

Regarding the first variant in which the pursuer's goal is to find (establish visibility with) an evader, in [13,12], the authors developed a complete algorithm for computing a successful motion strategy for a single pursuer with an unbounded field of view and range. The authors also establish the minimum number of pursuers to complete the task for simply connected free spaces. In [11], the authors present a complete algorithm for finding an evader with a pursuer who has a limited field of view. In [34], the authors address this first variant, in which the pursuer tries to find the evader inside a simply connected polygonal environment, and the evader, in turn, tries to avoid detection. They present a study in which bounds on the speeds of the pursuer and evader are given. In [16] the authors also address the first variant, and they pointed out that pursuit-evasion problems correspond to a broad and complex class of problems. Those problems have been modeled using optimal control for continuous differential games or graphs and algorithms for discrete pursuit-evasion. The first modeling often does not deal with obstacles, and the second one ignores dynamic constraints. The authors propose an algorithmically feasible solution yielding a stochastic hybrid controller. In this paper, we also want to address both obstacles and dynamic constraints. However, our proposed solution is not based on a stochastic hybrid controller as in [16] but on optimal sampling-based motion planning [19,18] and optimal control [2,28]. Moreover, the studied problem in this paper is different from the one in [16].

More recently, [33] also deals with a searching task. In the beginning, the pursuer cannot see the evader, and its job is to determine a path to find it, putting the evader inside its field of view. In particular, in [33] the authors established an upper bound on the positioning error that the pursuer may experience that still guarantees to find the evader. Again, the problem in [33] is different from the one presented in this paper, here at the beginning of the game, the pursuer can see the evader, and its task is to maintain that visibility.

The problem addressed in the present work corresponds to the second variant. The pursuer wants to maintain an evader's visibility who wants to escape from it, for instance, hidden behind an obstacle. Several works have addressed such a game; however, it *has not been completely solved yet.* Those works have different simplifications and assumptions, for instance, holonomic robots, point robots, a single corner analysis, etc. In this paper, we are addressing a complex problem. The players are nonholonomic robots shaped like discs in an environment with several obstacles representing multiple possibilities to escape. We also consider a case in which the pursuer's visibility region is limited by range.

Among the works mentioned above, in [7] the authors address this second variant in which the pursuer and evader are holonomic agents modeled as points. The authors analyze this problem as a game of kind (deciding which player wins). They use an explicit policy method to compute guaranteed surveillance strategies for the pursuer in an environment containing a single corner. These strategies depend on the initial positions of the pursuer and the evader in the workspace. In [8], the authors present a game-theoretic analysis of the problem in an environment containing obstacles. The pursuer and the evader are holonomic agents having bounded speeds. Both players have a complete map of the environment, omnidirectional vision and know each other's current position as long as they are visible between them. Using the Isaacs methodology, strategies for the players that are in the Nash Equilibrium are provided. Note that the authors assume a termination situation in which the evader escapes using a corner obstacle in that work. In this paper, we are mainly interested in the case in which the pursuer can maintain the evader's visibility forever.

In [39] the authors proposed an analytical solution and a dominant strategy for a pursuer to maintain the visibility of an evader considering a single corner. However, conversely to our modeling, in that work, both players are considered as holonomic points. Moreover,

in the present paper, several corners are considered simultaneously, generating multiple potential escape regions. In [38], the same authors from [39] compute approximations of solution sets for a visibility-based pursuit-evasion game. As in this work, the authors deal with nonholonomic constraints. They analyze three dynamical systems: a Dubins car, a Reeds-Shepp car, and a Differential Drive Robot (DDR). In contrast with the present work in [38], the authors do not consider unknown environments nor limited sensor range.

Also, [37] deals with the problem of moving a camera to maximize the visibility of a group of targets. Their proposed algorithm is based on the notion of visibility integrity, used to build the so-called visibility integrity roadmap (VIR) queried to plan a surveillance strategy. Both [37] and the present paper have sampling-based elements. Nonetheless, [37] might utilize sampling to construct the VIR and to generate new camera and predicted targets' positions, while in our method, a sampling-based planner – the RRT* – is employed to compute the shortest paths to escape and prevention from escape regions. Two additional differences to [37] are that this work addresses nonholonomic robotic systems and the possibility of using the presented method to retrieve motion strategies for both the pursuer and the evader. On the other hand, [37] is a multi-target method, while our approach focuses on a single evader. Additionally, the present work seeks to provide an analysis of surveillance guarantees.

Interestingly, other works have mixed the problem of maintaining evader's visibility with complementary tasks. For instance, it is possible to imagine a scenario in which the pursuer first looks for the evader; once it has established visibility with the evader, it tries to keep surveillance of it; if the evader escapes, the pursuer relaunches the finding strategy, and so on [21]. Another example of complementary tasks combined with the evader's tracking problem is the work in [5], where an agent is assigned the task of patrolling a grid-like environment in addition to keeping surveillance of an adversarial agent. The surveillance task is modeled as a temporal logic constraint. The whole patrolling task is efficiently implemented through computing information gains in vantage points utilizing a convolutional neural network.

Another mixed pursuit-evasion game is studied in [36]. In that work, the authors introduce the problem of planning a trajectory for an agent exploring an environment while avoiding being detected by an adversarial opponent. A game-theoric approach is adopted to solve it. The problem is modeled as a discrete, sequential, two-players, zero-sum game. The agent receives a positive reward for exploring new regions, and a negative penalty is granted every time the opponent detects the agent. To compute a solution, the authors use two types of tree search algorithms, minimax, and Monte-Carlo, on a grid-based discretization of the environment. For reducing the computational time, the authors propose pruning strategies that preserve optimality. Two significant differences between the work in [36] and our current work are the following. In [36], the solution is based on a grid-based discretization of the environment. In contrast, our method is based on the changes in the pursuer's visibility region and its polygonal representation. Another difference is how the motion strategies of the players are computed. In [36], the authors propose an approach based on tree search algorithms, while in our work, they are obtained using sample-based techniques.

In the third variant, probably the oldest one which comes from the control community [15,1,10], the pursuer wants to capture the evader, that is, move to a contact configuration, or closer than a given distance [15,30], and the evader wants to avoid to be captured. The classical Isaacs methodology [15] is a commonly used tool to address such a game. We stress that the proposed formulation in the present paper is novel compared to the modeling in classical differential games [15], since, in the addressed problem, there is no clear termination condition for the case when the pursuer can maintain the evader's visibility forever. In [17] the authors consider a two-player differential game in which the evader's objective is to reach in minimum time a goal set. At the same time, the pursuer tries to capture the evader (that is, to get closer to it than a given distance) before the evader reaches the goal set. The authors propose a sampling-based algorithm to compute optimal open-loop strategies for the evader, assuming a worst-case behavior for the pursuer. In this work, we also use a sampling-based algorithm to calculate optimal trajectories for the players. However, the game that we address is other, differently to [17], the pursuer's objective is to maintain the evader inside its field of view. In contrast, the evader tries to escape the surveillance of the pursuer by getting out of the pursuer's field of view or breaking visibility by hiding behind an obstacle.

In the vein of mixed pursuit-evasion, a game combining the search and capture problems in polygonal environments is studied in [24]. The players can have access to each other's locations only if the line connecting their current locations lies entirely inside the environment, i.e., if they can see each other. In that work, the authors represent the game's state using a visibility-based decomposition of the environment. The optimal players' strategies are computed using a min-max search algorithm. If the players are not visible at

the beginning of the game, first, the pursuer generates a plan to find the evader. Once that task has been accomplished, the pursuer computes a strategy to capture it. One difference between [24] and our current work is that in the first one, the authors consider holonomic players translating in the environment. In contrast, in our case, the players are non-holonomic robots. Another difference is that in [24], the environment is a simply connected polygon, while in our case, the environment can also have holes. An additional significant difference is how the strategies of the players are computed. In [24], they use a min-max search tree algorithm, while in our case, we use a sampling-based approach.

1.2 Main contributions

This paper presents several new contributions. The most important of them are three, the first two of theoretical nature and the third of experimental nature.

1. A new formulation for modeling and solving the visibility-based pursuit-evasion game, that yields a pursuer's strategy that considers the different shortest trajectories that the evader can follow to reach its time-variant goal. The evader's time-variant goal results from the evolution of the pursuer's visibility region. Moreover, there is no termination condition in this original formulation when the pursuer wins, since it can maintain visibility forever. Hence, it is not clear how to use classical methodologies as the Isaacs one.
2. A sufficient condition to maintain the evader's visibility; this condition holds for the case when the evader reacts irrationally.
3. A model general enough that considers realistic aspects of the game, such as unknown environments, the players' irrationality, nonholonomic constraints in the players, and limited sensor range. Thus it can potentially be applied in some practical problems, for instance, cars in airports and supermarkets that automatically follow a human being or convoys of autonomous vehicles.

The remainder of the paper is organized as follows. Section 2 formalizes our problem formulation. Section 3 presents a sufficient condition to keep surveillance and provides some results when the evader acts irrationally. Section 4 describes the proposed approaches for globally and locally known environments. Also, the case of a pursuer with a visibility region limited by range is presented. Section 5 provides a unified description of the surveillance algorithms proposed in the paper. Some simulations of the algorithms are presented in Section 6, and Section 7 concludes our work.

2 Problem formulation

Two disc-shaped Differential Drive Robots (DDRs), with the same radius, equipped with omnidirectional sensors, move in a 2D environment with obstacles. Without loss of generality, one of the DDRs is called the pursuer, and the other is called the evader. The pursuer's goal is to maintain the evader's visibility at all times while the evader wants to avoid surveillance. The pursuer sees the evader if a clear line of sight connecting the circular robots' center does not intersect an obstacle. Our work is focused on finding the motion strategies for the players to achieve only a single pair of goals, i.e., considering a single visit to a pair of escape and prevention from escape regions, in contrast, to compute a sequence of motions between pair of goals. The problem formulation is presented in a general manner, and in subsequent sections, it is further specialized to address different aspects of the problem.

2.1 Kinematic model

This work assumes that both players are equipped with unitary size wheels without loss of generality. Let $\mathbf{x}_p(t) = (x_p, y_p, \theta_p)$ and $\mathbf{x}_e(t) = (x_e, y_e, \theta_e)$ represent the pose of the pursuer and the evader at a given time instant t, more precisely, the pose of the centres of the discs that model the players. The following motion equations describe the evolution of the system

$$\begin{aligned} \dot{x}_p &= \left(\frac{u_1+u_2}{2}\right)\cos\theta_p, \quad \dot{x}_e = \left(\frac{v_1+v_2}{2}\right)\cos\theta_e, \\ \dot{y}_p &= \left(\frac{u_1+u_2}{2}\right)\sin\theta_p, \quad \dot{y}_e = \left(\frac{v_1+v_2}{2}\right)\sin\theta_e, \\ \dot{\theta}_p &= \left(\frac{u_2-u_1}{2b}\right), \quad \dot{\theta}_e = \left(\frac{v_2-v_1}{2b}\right), \end{aligned} \tag{1}$$

where $u_1, u_2 \in [-V^{\max}, V^{\max}]$ are the velocities of the left and the right wheel of the pursuer, respectively. Analogously, $v_1, v_2 \in [-V^{\max}, V^{\max}]$ are the velocities of the left and the right wheel of the evader, respectively. Parameter b is the distance between the center of the robot and the wheel location. We further define $\mathbf{v} = [v_1, v_2]^{\mathbf{T}}$, for later use. Note that since the wheels have a unitary radius, their translational and rotational speeds are equivalent. The players' heading angles, θ_p and θ_e, are measured in counter-clockwise sense from the positive x-axis.

2.2 Visibility-based formulation

To simplify the analysis, we assume that the game takes place in a polygonal environment $\mathbf{P}$ with obstacles;

however, the proposed approach can be extended to other types of environments, for instance, curved environments. The visibility region of the pursuer at pose $\mathbf{x}_p(t)$ is denoted by $V(\mathbf{x}_p(t))$, and the visibility region of the evader at pose $\mathbf{x}_e(t)$ is denoted by $V(\mathbf{x}_e(t))$. Note that the visibility regions of the players may change as they travel in the environment.

The goal of the pursuer is to maintain visibility of the evader at all time, i.e., $\mathbf{x}_e(t) \in V(\mathbf{x}_p(t)) \ \forall \ t$, while the evader wants to escape from $V(\mathbf{x}_p(t))$. Each edge of $V(\mathbf{x}_p(t))$ is either the border of an obstacle or the free space. We denote the edges that border the free space as *free edges* (see Fig. 1). Those edges are the ones that the evader needs to cross to escape from $V(\mathbf{x}_p(t))$ and define the *escape regions*, i.e., the regions where the evader can avoid surveillance. Each free edge in the visibility region of $V(\mathbf{x}_p(t))$ is generated by a *reflex vertex*[1] in the environment. Let v_i be a reflex vertex, and $l_i(t)$ the free edge generated by v_i when the pursuer is at $\mathbf{x}_p(t)$. Note that if the pursuer moves on the environment, then its visibility region changes, so do the free edges defining the region. Before presenting the formal definition of an escape region, let us introduce some notation. Let A be any set, then, A° will denote its interior, $\overline{A}$ its closure, and ∂A its boundary.

Definition 1 An *escape region*, $E_i(t)$, is the connected subset of $\overline{\mathbf{P}^\circ \setminus V(\mathbf{x}_p(t))}$, such that $l_i(t) \subset \partial E_i(t)$.

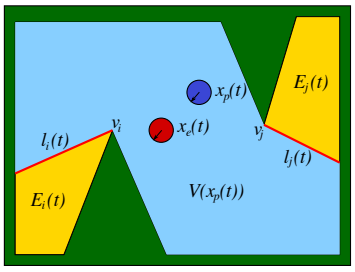


Fig. 1: Escape regions for the evader. The visibility region of the pursuer $V(\mathbf{x}_p(t))$ is shown in blue. There are two escape regions $E_i(t)$ and $E_j(t)$ (shown in gold) for the evader when the players are at the shown configurations. The corresponding free edges $l_i(t)$ and $l_j(t)$ are represented by the red lines.

The pursuer follows a strategy to maintain surveillance in which it moves from its current position to critical regions in the free space that prevents the evader's escape. Those *prevention regions* are obtained by extending rays from all reflex vertices in the environment, as shown in Fig. 2. More precisely, for each reflex vertex v_i, a ray is infinitely extended from v_i toward $\mathbf{P}^\circ$, such that the ray is collinear to the supporting line of an environment's edge incident to v_i. Following that manner, two rays are obtained per reflex vertex v_i; refer to them as $r_1(v_i)$ and $r_2(v_i)$. A prevention region is defined as follows:

Definition 2 Let R_i be the connected subset of $\mathbf{P}^\circ$ delimited by $r_1(v_i)$ and $r_2(v_i)$, such that $v_i \in \partial R_i$. A *prevention region*, P_i, is $\overline{R_i}$.

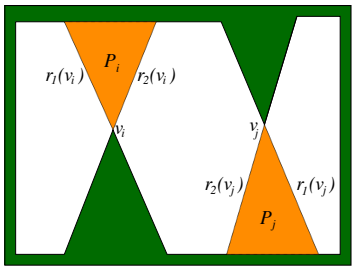


Fig. 2: Prevention regions for the pursuer. The reflex vertices v_i and v_j generate two prevention regions P_i and P_j shown in orange.

As it is shown in Proposition 1, there is a one-to-one correspondence between the escape regions (bounded by free edges) for the evader and the prevention regions for the pursuer. If the evader decides to escape through a free edge $l_i(t)$ and reach the corresponding escape region, $E_i(t)$, then to maintain surveillance the pursuer needs to reach on time the prevention region P_i. Thus, let $\tau(R, \mathbf{x})$ be the time that will take to a given player to reach a region R, travelling trajectory $\mathbf{x}$, with start pose $\mathbf{x}^0$ and duration T. If $\mathbf{x}(T) \in R$, then $\tau(R, \mathbf{x}) = T$, otherwise, $\tau(R, \mathbf{x}) = \infty$, with abuse of notation ∞ representing a very large number.

Proposition 1 *Let* $I = \{i : v_i \in V(\mathbf{x}_p(t)) \wedge \mathbf{x}_p(t) \notin P_i\}$. *There is a one-to-one correspondence between the escape regions,* $E_i(t)$, *and the prevention regions,* P_i, *provided* $i \in I$.

Proof By construction (Definition 1), there is a bijection between the free edges, $l_i(t)$, and escape regions, $E_i(t)$. Moreover, given $i \in I$, for the reflex vertices, v_i, there is also a bijection between them and the free edges, since each $l_i(t)$ is the border between $V(\mathbf{x}_p(t))$ and the free space resulting from the visual occlusion produced by v_i. On the other hand, by construction (Definition 2), there is a bijection between reflex vertices, v_i, and prevention regions, P_i, since each P_i is defined by rays $r_1(v_i)$ and $r_2(v_i)$. Thus, since there is a one-to-one correspondence between each v_i and each

[1] A reflex vertex in a polygon has an internal angle greater than π.

$E_i(t)$, with $i \in I$, and between each v_i and each P_i, then, provided $i \in I$, there is also a one-to-one correspondence between $E_i(t)$ and P_i.

Our modeling considers that updated information about the current players' poses is available or can be estimated for both the evader and the pursuer, every T_ϵ time units. Based on that information, the players will compute their optimal motion strategy and execute the resulting plans for at most T_ϵ time units, that is, until the next updated player's poses are received. Every time the poses' information is updated, a new plan is computed and executed by the players. It is assumed that the time that the players take to compute their strategies is negligible.

As mentioned before, it is considered that the players compute *one step ahead strategies*, namely, at a given time instant t, they plan only considering the subset of reflex vertices that are in $V(\mathbf{x}_p(t))$, refer to that subset as $\mathbb{V}$. They plan considering a single visit ahead to a pair $(P_i, E_i(t))$, conversely to planning for a sequence of visits to pairs of escape and prevention regions. Thus, it is considered that it is suitable for the evader to move to the region $E_i(t)$ that is more promising for it to escape, and for the pursuer to move to the respective region P_i to prevent the escape. Such players' behaviors define what we refer to as a rational evader and a rational pursuer. The fact that the player's pose information is received every T_ϵ time units and the motion strategies are updated accordingly will allow accounting for deviations from the optimal plans. In that manner, each player will react to the other player's irrational behavior.

Considering the setting mentioned above, in the planning stage, we address the next pursuit-evasion problem:

Problem 1 (*Visibility Pursuit-Evasion Problem*) Given start poses $\mathbf{x}_e^0$ and $\mathbf{x}_p^0$, let $\mathbf{x}_p^* = \arg\min_{\mathbf{x}_p}\{\tau(P_i, \mathbf{x}_p)\}$, with $\mathbf{x}_p^*(0) = \mathbf{x}_p^0$ and T_p^* the duration of $\mathbf{x}_p^*$. Also, let $E_i(t)$ evolve according to $\mathbf{x}_p^*$. Then, compute the trajectory $\mathbf{x}_e$, with $\mathbf{x}_e(0) = \mathbf{x}_e^0$, and the reflex vertex $v_i \in \mathbb{V}$, such that they optimize the next equation

$$\Psi = \max_{v_i \in \mathbb{V}}\{\max_{t\in[0,T_p^*]}\{t - \min_{\mathbf{x}_e}\{\tau(E_i(t), \mathbf{x}_e)\}\}\}. \quad (2)$$

The rationale behind Equation (2) is as follows. Ψ is the difference between the time needed for the evader to escape and the time needed for the pursuer to prevent the evader's escape under the players' optimal control. First, notice that the escape region, $E_i(t)$, varies as a function of time as the pursuer moves, conversely to the prevention region, P_i, which remains static given a reflex vertex v_i (see Fig. 3). Based on such observation, the *minimum time trajectory* $\mathbf{x}_p^*$ from $\mathbf{x}_p^0$ to P_i, is first computed. It is then assumed that $E_i(t)$ will vary according to the pursuer's motion that follows $\mathbf{x}_p^*$. This conforms a series of potential goals, $E_i(t)$, given a single vertex v_i. The different goals give rise to a series of minimum time planning problems for the evader that needs to be solved independently. The minimum time problems to reach different escape regions, e.g., $E_i(t_1)$ and $E_i(t_2)$, with $t_1 \neq t_2$, might yield disjoint trajectories that only have in common the initial state $\mathbf{x}_e^0$ (see Fig. 3). The second max operator in Equation (2) is in charge of iterating over each of those different planning problems for the evader, defined by each $t \in [0, T_p^*]$, selecting the worst case for the pursuer given a vertex v_i. Also notice that each $t \in [0, T_p^*]$ represents the minimum time for the pursuer to reach any intermediate pose, $\mathbf{x}_p^*(t)$, toward reaching P_i, moreover, there is a $E_i(t)$ related to each $\mathbf{x}_p^*(t)$. The subtraction operation in Equation (2) compares pursuer's related times and evader's times. Finally, the outermost max operator selects the most promissory vertex to produce an evader's escape. In the next section, we present a sufficient condition to keep surveillance based on the subtraction mentioned above between the players' computed times.

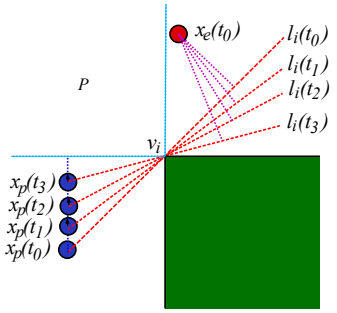


Fig. 3: Evolution of the free edge l_i defining the evader's escape regions as the pursuer travels to the corresponding prevention region. The blue dotted line shows the pursuer's trajectory to reach the prevention region. The magenta dotted lines show the evader's trajectories to reach the corresponding free edge (red dotted lines) as the pursuer moves. Note that those trajectories are for illustration purposes and are assumed to be the ones that make the players travel their minimum time trajectories to reach their objectives.

Below, in Proposition 4, we will prove that our formulation does not imply any cooperation between the players.

3 Sufficient Condition for Surveillance

In this section, a sufficient condition to keep surveillance is provided first. It is followed by an analysis of

the time window that the pursuer needs to identify that re-planning is needed– pursuer's reaction time–to keep still surveillance of an evader that might have acted irrationally (deviated from its optimal strategy). The sufficient condition is provided through Proposition 2. The result concerning the bounds on reaction time for the pursuer is provided in Proposition 3, aided by Lemma 1. That lemma presents a result in which it is stated that, under certain conditions, the pursuer can remain static during a time and delay the start of its motion but can still maintain the evader's surveillance. Proposition 3 uses that lemma to establish a strategy called Γ, on which its proof is built.

Proposition 2 *Given a reflex vertex $v_i \in \mathbb{V}$ that has related to it a potential escape of the evader, let $\mathbf{x}_p^* = \arg\min_{\mathbf{x}_p}\{\tau(P_i, \mathbf{x}_p)\}$, and T_p^* the duration of $\mathbf{x}_p^*$. Let $E_i(t)$ evolve according to $\mathbf{x}_p^*$. If $\{t - \min_{\mathbf{x}_e}\{\tau(E_i(t), \mathbf{x}_e)\}\} \leq 0$, $\forall t \in [0, T_p^*]$, then the pursuer prevents an evader's escape using v_i.*

Proof This proposition is proved by contradiction. Let us assume that $\{t - \min_{\mathbf{x}_e}\{\tau(E_i(t), \mathbf{x}_e)\}\} \leq 0$, $\forall t \in [0, T_p^*]$, but that there exists a time $t \in [0, T_p^*]$ at which the evader escapes. That would mean that there is a time $\min_{\mathbf{x}_e}\{\tau(E_i(t), \mathbf{x}_e)\}$ for the evader to reach its respective goal, smaller than the time t that takes to the pursuer to take the free edge to the state $l_i(t)$, that is, $\exists t \in [0, T_p^*]$ s.t. $\{t - \min_{\mathbf{x}_e}\{\tau(E_i(t), \mathbf{x}_e)\}\} > 0$. However, since it has been assumed that $\{t - \min_{\mathbf{x}_e}\{\tau(E_i(t), \mathbf{x}_e)\}\} \leq 0$, $\forall t \in [0, T_p^*]$, a contradiction occurs, the result follows.

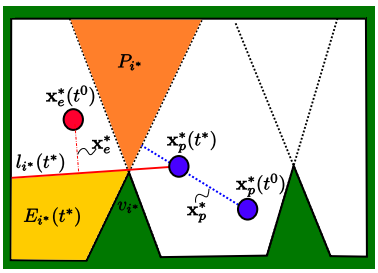


Fig. 4: A sample graphical representation of the solution to Equation (2), including elements as the vertex v_{i^*} that sets the worst case for the pursuer, the optimal trajectories for the players, $\mathbf{x}_p^*$ and $\mathbf{x}_e^*$, the time t^* that sets the worst state for the free edge, $l_{i^*}(t^*)$, and the related escape region $E_{i^*}(t^*)$.

From Equation (2), it can be seen that Ψ encodes the worst-case scenario for the pursuer, more precisely, the vertex v_i and time t that defines an evader's behavior that is more difficult for the pursuer to keep surveillance of. Refer to the index of such a vertex as i^*, and as t^* to such time (the arguments that solve Equation (2), see Fig. 4). Ψ is the difference between the time it takes for the pursuer to bring the free edge to state $l_{i^*}(t^*)$, and for the evader to reach $E_{i^*}(t^*)$. It defines a time margin of the pursuer's advantage over the evader. Since that is the worst case for the pursuer, that time margin Ψ is a lower bound for any pair $l_i(t)$ and $E_i(t)$. Taking this into account, we present Lemma 1 and Proposition 3.

Lemma 1 *If $\Psi < 0$, then there exists an idle time $T_I \leq |\Psi|$ for the pursuer, during which it can apply no action and still be able to prevent an escape of the evader using v_{i^*}.*

Proof Term Ψ defines a pursuer's time advantage over the evader, where $\Psi = t^* - \tau(E_{i^*}(t^*), \mathbf{x}_e^*)$. Consider that the pursuer stays idle for a time period $T_I < |\Psi|$, and starts traversing its optimal trajectory at time $t = T_I$. Under that setting, the actual time that would take for the pursuer to bring the free edge to state $l_{i^*}(t^*)$, would be $t + t^*$ time units. The true time margin for the pursuer is

$$\begin{aligned}\hat{\Psi} &= t^* + t - \tau(E_{i^*}(t^*), \mathbf{x}_e^*) \\ &= t + \Psi.\end{aligned}$$

Assume the worst case where the idle time $T_I = |\Psi|$, hence, $t = |\Psi|$. Consequently, $\hat{\Psi} = |\Psi| + \Psi$, which results into $\hat{\Psi} = 0$, meaning that the pursuer could still maintain evader's surveillance. The result follows.

Proposition 3 *Assume that, according to Equation (2), at instant t^0, the players compute their optimal strategies, but the evader does not follow its optimal one. Provided that $\Psi < 0$, there exists an upper bound on a reaction time $T_R > 0$, s.t. the pursuer can still prevent a one-step-ahead evader's escape by correcting its trajectory at instant t, in which $t^0 \leq t \leq t^0 + T_R$.*

Proof Suppose at time t^0, the pursuer executes its optimal trajectory towards P_{i^*}, but the evader deviates from its optimal strategy. In that case, the pursuer might not only be trying to address an escape through a wrong vertex v_{i^*}, but it can even cooperate to the evader's escape (see Fig. 5). Let T_c be the infimum on the time that will take the evader to escape considering that the pursuer will follow the trajectory that favors the escape the most. Note that $T_c > 0$, since the escape has not already taken place.

The fact that $\Psi < 0$ implies $|\Psi| > 0$, which is a lower bound on the time margin of the pursuer's advantage over the evader to prevent a one step ahead escape using any $v_i \in \mathbb{V}$. If $|\Psi| > 0$ and $|\Psi| \leq T_c$, then the pursuer can return to pose $\mathbf{x}_p(t^0)$ without losing sight of the evader. Refer to such correction strategy of the pursuer

going back to $\mathbf{x}_p(t^0)$, as Γ. If an escape does not take place, returning to $\mathbf{x}_p(t^0)$ is equivalent to the pursuer staying idle for the time it took it to return to $\mathbf{x}_p(t^0)$. From Lemma 1, there exists an idle time $T_I \leq |\Psi|$ for the pursuer that allows it to still prevent an escape through what was v_{i^*} at t^0, consequently, through any other $v_j \neq v_{i^*}$. A bound on reaction time $T_R < |\Psi|/2$ will allow the pursuer to react on time, apply strategy Γ, and once at $\mathbf{x}_p(t^0)$, recompute its strategy through Equation (2) contemplating the new evader's pose.

If $|\Psi| > 0$ and $T_c < |\Psi|$, then a bound on reaction time considering $T_R < T_c/2$ would allow the pursuer to apply strategy Γ and correct its plan. Therefore, $T_R < \min\{|\Psi|/2, T_c/2\}$, and noting that both $|\Psi| > 0$ and $T_c > 0$, then $T_R > 0$. The result follows.

Remark 1 The reaction time t, in which $t^0 \leq t$, depends on how often the knowledge of $(\mathbf{x}_p(t), \mathbf{x}_e(t))$ is available to the players; hence, $t = t^0 + T_\epsilon$. Thus, $T_\epsilon \leq T_R$ is needed for the pursuer to react on time to evader's deviations.

Remark 2 The strategy Γ is proposed only to prove Proposition 3. However, in reality, at time t, in which $t^0 \leq t \leq t^0 + T_R$, the pursuer can recompute Equation (2) with the updated poses, obtaining in that manner a new trajectory that outperforms the one that requires the execution of strategy Γ.

4 Surveillance Algorithms

The proposed methodology to address Problem 1 is presented next. Three approaches that address different aspects of the game are described. The first one assumes that the environment $\mathbf{P}$ is known. In the second one, the players only have local representations of $\mathbf{P}$, that is, the players compute their motions strategies solely using $V(\mathbf{x}_p(t))$ and $V(\mathbf{x}_e(t))$. In both approaches, the sensor range is assumed unlimited. The third approach considers that the environment is known, but the sensor range is limited. Before presenting the approaches mentioned above, we start by providing the details on how the minimum time trajectories are computed for DDRs, which aim at minimizing $\tau(R, \mathbf{x})$, and which play a key role in our methodology.

4.1 Minimum Time Trajectories

The computation of trajectories $\mathbf{x}_e$ and $\mathbf{x}_p$ will be done through the RRT* sampling-based methodology [19, 18, 20]. Independent trees will be constructed for each

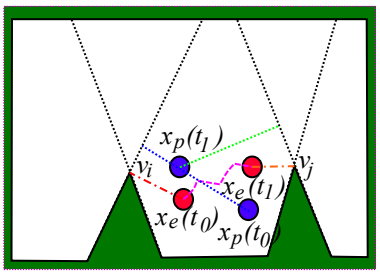


Fig. 5: The evader follows a sub-optimal strategy but the pursuer is able to react to it. The initial pose of the pursuer is $\mathbf{x}_p(t_0)$ and the evader's initial pose is $\mathbf{x}_e(t_0)$. In this case, the evader's optimal strategy at pose $\mathbf{x}_e(t_0)$ corresponds to reach vertex v_i following its minimum time trajectory. Instead, the evader follows the magenta dotted curve reaching pose $\mathbf{x}_e(t_1)$, where the evader's optimal strategy changes to reach vertex v_j. The pursuer follows its optimal strategy at $\mathbf{x}_p(t_0)$ moving to vertex v_i reaching position $\mathbf{x}_p(t_1)$. After reaching $\mathbf{x}_p(t_1)$, the pursuer reacts to the change of strategy of the evader and starts moving to vertex v_j.

of the players to solve motion planning queries. Notice that the considered robots are nonholonomic systems, hence, the RRT* methodology will be applied in the context of kinodynamic planning. From [18], to use the RRT* in the aforesaid context, a distance function along with a steering procedure that respects the system's differential constraints must be proposed. To facilitate the presentation, only the evader's equations will be developed but the pursuer's ones are equivalent. The distance function between any two poses z_1 and z_2, must have the form $\text{dist}(z_1, z_2) = \min_{T \in \mathbb{R}_{\geq 0}, \mathbf{v}} J(\mathbf{x}_e)$, s.t. $\dot{\mathbf{x}}_e = f(\mathbf{x}_e, \mathbf{v})\ \forall t$, with $J(\mathbf{x}_e) = \int_0^T c(\mathbf{x}_e(t))dt$, and $\mathbf{x}_e(0) = z_1$, $\mathbf{x}_e(T) = z_2$. Since we seek to minimize the travelled time for each of the players, the cost $c(\cdot)$ is set to 1 [4].

Concerning the steering procedure, the planner presented in [2] will be used in the construction of the trees. Such a planner computes time-optimal trajectories between any two poses for a DDR in the absence of obstacles. In [2], it was found that the optimal motion primitives are either rotations in site (clockwise or counter-clockwise rotations) or straight-line motions (forward or backward motions), or their optimal concatenation.

4.2 Known Environment

Here, we present our approach to solving Problem 1 considering that the whole map of the environment $\mathbf{P}$ is known for both players. It is also assumed that they receive information about both players' poses every T_ϵ units of time. Thus, each player can compute its visibility region and the visibility region of the other player.

Equation (2) describes the max-min approach used to model the game. Given a reflex vertex $v_i \in \mathbb{V}$, the modelling implies that each player needs to find the time-optimal trajectory to reach its respective sub-goal, namely, P_i or $E_i(t)$. For computing the optimal trajectories, the methodology in Section 4.1 is used. Notice that the RRT* is asymptotically optimal. However, for a sufficiently large number of nodes, we can expect that the best trajectories encoded in the pursuer's and evader's trees, $\hat{\mathbf{x}}_p$ and $\hat{\mathbf{x}}_e$, will be sufficiently good approximations to the optimal trajectories. The RRT* algorithm will consider the poses received in the last information update, $\mathbf{x}_p^0$ and $\mathbf{x}_e^0$, as the player's start poses, hence, $\hat{\mathbf{x}}_p(0) = \mathbf{x}_p^0$ and $\hat{\mathbf{x}}_e(0) = \mathbf{x}_e^0$; however, the algorithm also requires the definition of a goal set, X_{goal}, for each player.

Concerning the pursuer's tree, the goal set X_{goal} is simply fixed to P_i. For the evader's tree, recall that $E_i(t)$ varies as a function of time as the pursuer moves. This phenomena makes the proposal of X_{goal} for the evader a non-trivial task. To address such evolution of $E_i(t)$, when the players are at $\mathbf{x}_p^0$ and $\mathbf{x}_e^0$ planning their motions given a reflex vertex v_i, several queries are computed for the evader. More precisely, the pursuer's trajectory $\hat{\mathbf{x}}_p(t)$ that approximates the minimum time trajectory to reach P_i, is first computed; that fixes $E_i(t)$, which evolves as the pursuer traverses $\hat{\mathbf{x}}_p(t)$. Then, the time interval $[0, \tau(P_i, \hat{\mathbf{x}}_p)]$ is split into M subintervals of duration Δt, hence, let $t_j = j \cdot \Delta t$, with $j = 0, 1, ..., M$.

Subsequently, a series of planning problems are solved for the evader, that is, for each t_j, the trajectory $\hat{\mathbf{x}}_e$ is computed using an RRT* to reach the goal set $E_i(t_j)$. Notice that a single RRT* can be queried to address all the planning problems toward goals $E_i(t_j)$, provided that the tree is sufficiently dense. Lastly, the trajectory $\hat{\mathbf{x}}_e$ that corresponds to the time t_j that maximizes $\{t_j - \min_{\hat{\mathbf{x}}_e}\{\tau(E_i(t_j), \hat{\mathbf{x}}_e)\}\}$, is stored. Such trajectory $\hat{\mathbf{x}}_e$ represents an approximation to the worst case for the pursuer given a reflex vertex v_i.

According to Equation (2), the vertex v_{i^*} that maximizes that equation in conjunction with the corresponding time t_j^*, fixes the pair of computed trajectories, $\hat{\mathbf{x}}_p^*$ and $\hat{\mathbf{x}}_e^*$, which will take the players toward P_{i^*} and $E_{i^*}(t_j^*)$. Rational players will move over $\hat{\mathbf{x}}_p^*$ and $\hat{\mathbf{x}}_e^*$ for a time period T_ϵ. Subsequently, no matter if the players moved rationally or not, new updated poses, $\mathbf{x}_p^0$ and $\mathbf{x}_e^0$, are received to proceed with the computation of optimal plans, repeating all the process again. If during the execution of motion, at some time t, $\mathbf{x}_e(t) \notin V(\mathbf{x}_p(t))$, then the game ends. Algorithm 1 implements the computation of the player's optimal strategies, which is done every time at which current poses, $\mathbf{x}_p^0$ and $\mathbf{x}_e^0$, are received.

Algorithm 1 Surveillance for Known Environments

Input: $\mathbf{P}$, $\mathbf{x}_p^0$, $\mathbf{x}_e^0$, n; //n is number of nodes in RRT*

1: $\hat{\mathbf{X}}_p \leftarrow \emptyset$; $\hat{\mathbf{X}}_e \leftarrow \emptyset$; $\hat{\mathcal{T}} \leftarrow \emptyset$;
2: $\mathbb{V} \leftarrow$ getSubset$(V(\mathbf{x}_p^0), \{v_i\})$;
3: **for all** $v_i \in \mathbb{V}$ **do**
4: $\quad P_i \leftarrow$ getPreventionRegion$(\mathbf{P}, v_i)$;
5: $\quad RRT_p^* \leftarrow$ getRRT$^*(\mathbf{x}_p^0, P_i, \mathbf{P})$;
6: $\quad \hat{\mathbf{x}}_p \leftarrow$ getBestTra(RRT_p^*, P_i);
7: $\quad \hat{\mathbf{X}}_p \leftarrow \hat{\mathbf{X}}_p \cup \hat{\mathbf{x}}_p$;
8: $\quad \{t_j\} \leftarrow$ getTimeDis$(\tau(P_i, \hat{\mathbf{x}}_p), M)$;
9: $\quad RRT_e^* \leftarrow$ getRRT$^*(\mathbf{x}_e^0, n, \mathbf{P})$;
10: $\quad \bar{\mathbf{X}}_e \leftarrow \emptyset$; $\bar{\mathcal{T}} \leftarrow \emptyset$;
11: $\quad$ **for all** $j \in \{0, 1, ..., M\}$ **do**
12: $\quad\quad l_i(t_j) \leftarrow$ getFreeEdge$(\mathbf{P}, \hat{\mathbf{x}}_p(t_j), v_i)$;
13: $\quad\quad E_i(t_j) \leftarrow$ getEscapeRegion$(\mathbf{P}, V(\hat{\mathbf{x}}_p(t_j)), l_i(t_j))$;
14: $\quad\quad \hat{\mathbf{x}}_e \leftarrow$ getBestTra$(RRT_e^*, E_i(t_j))$;
15: $\quad\quad \bar{\mathbf{X}}_e \leftarrow \bar{\mathbf{X}}_e \cup \hat{\mathbf{x}}_e$;
16: $\quad\quad \bar{\mathcal{T}} \leftarrow \bar{\mathcal{T}} \cup (t_j - \tau(E_i(t_j), \hat{\mathbf{x}}_e))$;
17: $\quad$ **end for**
18: $\quad j \leftarrow$ getArgMax$(\bar{\mathcal{T}})$;
19: $\quad \hat{\mathbf{X}}_e \leftarrow \hat{\mathbf{X}}_e \cup \bar{\mathbf{X}}_e[j]$;
20: $\quad \hat{\mathcal{T}} \leftarrow \hat{\mathcal{T}} \cup \bar{\mathcal{T}}[j]$;
21: **end for**
22: $i^* \leftarrow$ getArgMax$(\hat{\mathcal{T}})$;
23: $\hat{\mathbf{x}}_p^* \leftarrow \hat{\mathbf{X}}_p[i^*]$, $\hat{\mathbf{x}}_e^* \leftarrow \hat{\mathbf{X}}_e[i^*]$;
24: **return** $\hat{\mathbf{x}}_p^*$ and $\hat{\mathbf{x}}_e^*$;

Algorithm 1 starts by computing the subset of reflex vertices, $\mathbb{V}$, within the initial pursuer's visibility region, $V(\mathbf{x}_p^0)$ (line 2). From lines 3 to 21, each vertex $v_i \in \mathbb{V}$ is analysed. The analysis starts by computing the trajectory $\hat{\mathbf{x}}_p$ for the pursuer towards the goal set P_i. That is done from lines 4 to 7, computing an RRT* through procedure getRRT*$(\cdot)$, which computes the tree receiving an initial pose, a goal set, a fixed number of nodes, and the set representing the workspace. The resulting trajectory is stored in a data structure, $\hat{\mathbf{X}}_p$, where the trajectory can be retrieved by indexing it, e.g., $\hat{\mathbf{X}}_p[i]$. Line 8 is in charge of splitting the time interval $[0, \tau(P_i, \hat{\mathbf{x}}_p)]$ into M pieces to retrieve the set of times $\{t_j\}$. Line 9 computes an RRT* with n nodes from the evader's initial pose $\mathbf{x}_e^0$. From lines 11 to 17, that tree is queried to compute evader's trajectories toward goal sets $E_i(t_j)$; the trajectories are stored in data structure $\bar{\mathbf{X}}_e$. The worst-case for the pursuer given a vertex v_i is computed and stored in lines 18 to 20. Later on, line 22 retrieves the vertex index i^* that represents the worst case for the pursuer among all vertices $v_i \in \mathbb{V}$, that is, the most promissory free edge for the evader to escape. Line 23 retrieves the optimal trajectories for the players, $\hat{\mathbf{x}}_p^*$ and $\hat{\mathbf{x}}_e^*$, according to index i^*, and line 24 finally returns those trajectories. Notice that, since the players receive information about the pose of each other and have access to a map of $\mathbf{P}$, both players can run Algorithm 1 independently

and compute the pair $(P_{i^*}, E_{i^*}(t_j^*))$ to which they are supposed to move.

4.3 Unknown Environment

Generally speaking, Algorithm 1 also provides a sketch on how to proceed with the surveillance task when a global map of $\mathbf{P}$ is not at hand. In such case, it is again assumed that information about the current player's poses, $\mathbf{x}_p^0$ and $\mathbf{x}_e^0$, is received every T_ϵ time units, and that the only representations of $\mathbf{P}$ available to the players are the visibility regions $V(\mathbf{x}_p(t))$ and $V(\mathbf{x}_e(t))$, which serve as local representations of $\mathbf{P}$. Region $V(\mathbf{x}_p(t))$ is only known to the pursuer, and $V(\mathbf{x}_e(t))$ only known to the evader. Thus, it is suitable to plan using $\hat{\mathbf{P}} = V(\mathbf{x}_p(t)) \cap V(\mathbf{x}_e(t))$ in substitution to $\mathbf{P}$, which corresponds to the intersection between the player's visibility regions. Below, it is first proven that the set $\hat{\mathbf{P}} = V(\mathbf{x}_p(t)) \cap V(\mathbf{x}_e(t))$ can be computed independently by each player (refer to Proposition 4). Later, it is proved that assuming that both players will plan on the same workspace, namely, $\hat{\mathbf{P}} = V(\mathbf{x}_p(t)) \cap V(\mathbf{x}_e(t))$, does not represent any cooperation (refer to Proposition 5).

Proposition 4 *The set $\hat{\mathbf{P}} = V(\mathbf{x}_p(t)) \cap V(\mathbf{x}_e(t))$ can be computed independently by each player provided that $\mathbf{x}_e(t) \in V(\mathbf{x}_p(t))$, even when $V(\mathbf{x}_p(t))$ is unknown to the evader and $V(\mathbf{x}_e(t))$ is unknown to the pursuer.*

Proof Without loss of generality consider the computation of $\hat{\mathbf{P}}$ by the pursuer. Let $V(\mathbf{x}_p(t))$ be a first approximation of $\hat{\mathbf{P}}$. Subsequently, since $\mathbf{x}_e(t) \in V(\mathbf{x}_p(t))$, the pursuer can proceed to compute the set $\hat{V}(\mathbf{x}_e(t))$, as the set of points that are visible from $\mathbf{x}_e(t)$ that lie within $V(\mathbf{x}_p(t))$. Set $\hat{V}(\mathbf{x}_e(t))$ would be the set of points visible from both $\mathbf{x}_p(t)$ and $\mathbf{x}_e(t)$, namely, $V(\mathbf{x}_p(t)) \cap V(\mathbf{x}_e(t))$, which is indeed $\hat{\mathbf{P}}$. Equivalent arguments apply to the case when the evader does the computation. The result follows.

Proposition 5 *Assuming that all the vertices inside the set $\hat{\mathbf{P}} = V(\mathbf{x}_p(t)) \cap V(\mathbf{x}_e(t))$ will produce a possibility for the evader to escape, then the set $\hat{\mathbf{P}} = V(\mathbf{x}_p(t)) \cap V(\mathbf{x}_e(t))$ can be used to compute the players' strategies and this does not imply any type of cooperation between the players.*

Proof This proposition is proved by cases. There are only two cases that exhaustively consider all possibilities. Case 1 (see Fig. 7), the visibility region of the evader covers an area that is not covered by the visibility region of the pursuer. The evader must cross the boundary of the pursuer's visibility region to escape. Note that there might be vertices outside the pursuer's visibility region that the pursuer does not see and potentially can be used for the evader to escape, but those vertices are not relevant since the evader must first cross the boundary of the pursuer's visibility region to escape. Case 2 (see Fig. 8), the visibility region of the pursuer covers an area that is not covered by the evader's visibility region. In this case, the evader must consider all the vertices inside $\hat{\mathbf{P}} = V(\mathbf{x}_p(t)) \cap V(\mathbf{x}_e(t))$ to compute its motion strategy. There might be vertices that actually do not represent a possible escape since the pursuer from its position already sees both sides of those vertices. Nevertheless, since there is a possibility that a given vertex actually represents an escape, and the evader with the available information cannot be sure whether or not that is the scenario, then the evader shall compute its motion strategy considering all the potential possibilities to escape. Therefore, all relevant information is inside the intersection of the player's visibility regions; thus, using $\hat{\mathbf{P}}$ does not imply any cooperation between the players.

As a first consequence of working only with $V(\mathbf{x}_p(t))$ and $V(\mathbf{x}_e(t))$, the set $\mathbb{V}$ might contain reflex vertices that are not available for the evader to provoke a escape because the evader might not be aware of them; those are vertices $v_i \in \mathbb{V}$, such that $v_i \in V(\mathbf{x}_p(t))$ but $v_i \notin V(\mathbf{x}_e(t))$. Thus, the planning stage will only consider escapes related to reflex vertices that lie in the more restricted set $\hat{\mathbb{V}} = \{v_i : v_i \in \hat{\mathbf{P}}\}$.

Another main difference that arises w.r.t a globally known environment is the definition of goal sets, X_{goal}, used in the computation of trajectories $\hat{\mathbf{x}}_p$ and $\hat{\mathbf{x}}_e$ using the RRT*. A key aspect in the case of unknown environments is that one player's goal must be inferred by the other player with the information available at its current position. Regarding the pursuer's goal set, recall that for every reflex vertex v_i, two rays, $r_1(v_i)$ and $r_2(v_i)$, delimit a prevention region P_i (see Definition 2); nonetheless, there are cases in which for a vertex $v_i \in \hat{\mathbb{V}}$, one of the two supporting lines incident to v_i is not visible to one of the players, hence, while planning in $\hat{\mathbf{P}}$, four cases arise (see Fig. 6): (a) the two supporting lines of v_i are present in $\hat{\mathbf{P}}$, (b) one supporting line is present in $\hat{\mathbf{P}}$ and there is a free edge emerging from v_i related to the pursuer's visibility region, (c) one supporting line is present in $\hat{\mathbf{P}}$ and there is a free edge emerging from v_i related to the evader's visibility region, (d) no supporting line of v_i is present in $\hat{\mathbf{P}}$ and two free edges emerge from v_i, one from each of the players' visibility regions. Fig. 6 shows instances of the four scenarios. For case (a), the prevention region P_i might be computable. However, no free edge is generated for that case, so the corresponding vertex is discarded as an option to pro-

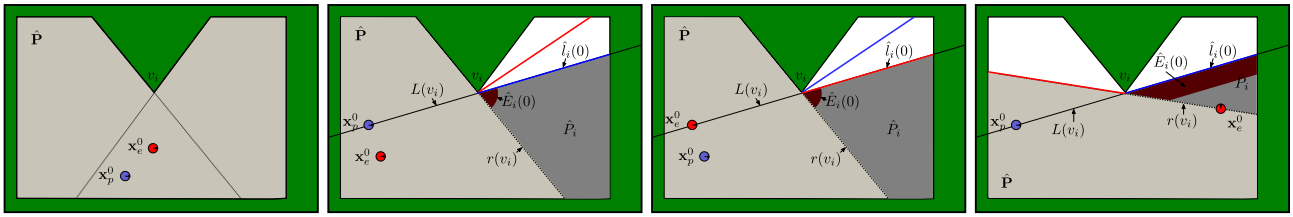


Fig. 6: Four cases that arise to define goal sets $\hat{P}_i$ and $\hat{E}_i(0)$. The workspace $\hat{\mathbf{P}}$ in light gray, set $\hat{P}_i$ in dark gray, and $\hat{E}_i(0)$ in dark red.

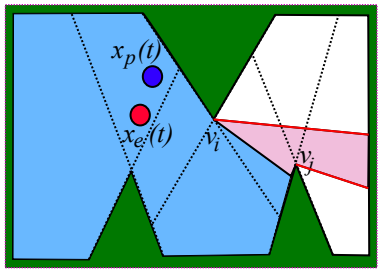


Fig. 7: The evader's visibility region (union of the light blue and magenta regions) covers an area not covered by the pursuer's visibility region (light blue). There might be vertices outside the pursuer's visibility region (see vertex v_j) that the pursuer does not see and potentially can be used for the evader to escape. However, those vertices are irrelevant since the evader must first cross the pursuer's visibility region's boundary to escape.

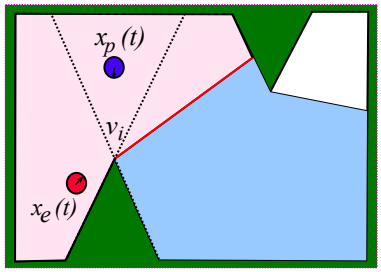


Fig. 8: The pursuer's visibility region (union of the light blue and magenta regions) covers an area not covered by the evader's visibility region (magenta region). The evader must consider all vertices inside the intersection of the pursuer's and evader's visibility regions to compute its motion strategy. There might be vertices that do not represent a possible escape (see v_i) since the pursuer from its position already sees both sides of those vertices. However, since there is a possibility that a given vertex actually represents an escape, and the evader with the available information cannot be sure whether or not that is the case, then the evader shall compute its motion strategy considering all the potential possibilities to escape.

duce an escape. For cases (b), (c), and (d), since there is at least one supporting line that is not in $\hat{\mathbf{P}}$, P_i is not computable. In those cases, the sub-goal region P_i is approximated by a region $\hat{P}_i$ as follows.

For case (b), see Fig. 6b, let the pursuer be at $\mathbf{x}_p^0$, and let $r(v_i)$ be the ray from v_i toward $\hat{\mathbf{P}}^\circ$, collinear to the supporting line of v_i that is visible from $\mathbf{x}_p^0$. Let $\hat{l}_i(0)$ be the initial free edge emerging from v_i in $\hat{\mathbf{P}}$, which in this case is a free edge resulting from $V(\mathbf{x}_p^0)$. Let $L(v_i)$ be a ray collinear to $\hat{l}_i(0)$ that infinitely extends from v_i toward $\hat{\mathbf{P}}^\circ$. Then, $\hat{P}_i$ is the closure of the connected subset of $\hat{\mathbf{P}}^\circ$ delimited by $r(v_i)$ and $L(v_i)$, such that $v_i \in \partial\hat{P}_i$. The rationale behind the construction of $\hat{P}_i$ is that, in the worst case, $r_1(v_i)$ and $r_2(v_i)$ will almost coincide at $r(v_i)$, hence, it is suitable to assume that $\hat{P}_i$ is delimited by a subset of $r(v_i)$. Nonetheless, if $r_1(v_i)$ and $r_2(v_i)$ coincide at $r(v_i)$, P_i would have measure zero, which is inconvenient for the RRT*. To overcome that issue, to generate the approximation $\hat{P}_i$ with non-zero measure, the set is further delimited by $L(v_i)$. Case (c) is similar to (b), with the only difference that $\hat{l}_i(0)$ in $\hat{\mathbf{P}}$ is a free edge related to the evader and not the pursuer, however, we proceed equally to compute $\hat{P}_i$.

Concerning case (d), see Fig. 6d, the procedure used to compute $\hat{P}_i$ cannot be completely the same as in cases (b) and (c) since the evader must be able to infer the pursuer's goal from its current position. Assuming the players are at $\mathbf{x}_p^0$ and $\mathbf{x}_e^0$, construct $r(v_i)$, emerging from v_i toward $\hat{\mathbf{P}}^\circ$, collinear to the line that joins v_i and $\mathbf{x}_e^0$. Once $r(v_i)$ has been constructed, proceed to build segment $L(v_i)$, thus, $\hat{P}_i$, just as it was done in cases (b) and (c). Notice that this is a conservative approach to build $\hat{P}_i$ since in general, $\hat{P}_i$ will be farther than P_i from $\mathbf{x}_p^0$, but if the pursuer can reach $\hat{P}_i$ on time to prevent an escape, then it will also be able to reach P_i on time.

Regarding the evader's goal set, it is worth noticing that since $\mathbf{P}$ is not available, the evolution of $E_i(t)$ according to $\hat{\mathbf{x}}_p$ cannot be computed. Moreover, there is information regarding the environment structure only at the time when the planning takes place, namely, $t = 0$. Hence, a single query is computed for the evader with $E_i(0)$ as the goal set, conversely to the case in Section 4.2 where several queries take place to account for the evolution of $E_i(t)$. Indeed, an approximation, $\hat{E}_i(0)$, is computed instead as follows.

Cases (b) and (c) will be first considered, and later, case (d) will be presented. Without loss of generality, let us focus on a reflex vertex $v_i \in \hat{\mathbb{V}}$, and assume that the players are at $\mathbf{x}_p^0$ and $\mathbf{x}_e^0$. Again, since $\hat{\mathbf{P}}$ has limited information, an approximation to the free edges, $\hat{l}_i(0)$, will be computed. Consider $\hat{l}_i(0)$ and $r(v_i)$ accordingly to each case as it was described before. Once $\hat{l}_i(0)$ and $r(v_i)$ have been identified, let $B(v_i, \eta)$ be a ball of radius η centred at vertex v_i. The goal set $\hat{E}_i(0)$ will be fixed to the closure of $\hat{\mathbf{P}} \cap B(v_i, \eta)$, delimited by $\hat{l}_i(0)$ and $r(v_i)$, considering η as twice the diameter of the disc that represents the evader (see Figs. 6b and 6c). Building $\hat{E}_i(0)$ in such a way, aims at generating an evader's goal set with non-zero measure around reflex vertex v_i, but sufficiently small to drive the evader toward v_i. For case (d) (see Fig. 6d) instead of defining a ball centered at vertex v_i, a band delimited by the free edge $\hat{l}_i(0)$ with width η is built; that band is the goal $\hat{E}_i(0)$ for the evader.

Notice that $\hat{E}_i(0)$ is always within $V(\mathbf{x}_p^0)$, but $\hat{E}_i(0)$ will drive the evader toward the boundaries of $V(\mathbf{x}_p^0)$. Once the evader is located at $\hat{E}_i(0)$, it is assumed that the evader can evaluate whether it can cross the free edge to produce an escape and execute the crossing. Analogously, it might happen that once the evader is at $\hat{E}_i(0)$, it realizes that no escape was possible using v_i. That is a consequence of planning under limited information, namely, planning using $\hat{\mathbf{P}}$.

Algorithm 2 summarizes the computation of the optimal trajectories when a global map of $\mathbf{P}$ is not available. The main differences compared to Algorithm 1 are the next ones. Line 2 computes the workspace $\hat{\mathbf{P}}$ that the planning stage will consider, and line 3 computes the set of reflex vertices $\hat{\mathbb{V}}$ that lie in $\hat{\mathbf{P}}$. Lines 4 to 15 only iterate for the reflex vertices in $\hat{\mathbb{V}}$ that do not fall in the scenario described in case (a). Lines 5, 9 and 10, compute the approximations to the pursuer's and evader's goals, $\hat{P}_i$ and $\hat{E}_i(0)$. Also, notice that the evader performs a single query for each reflex vertex, conversely to the multiple queries performed in lines 11 to 17 of Algorithm 1. Apart from that, Algorithm 2 proceeds equally to Algorithm 1.

Algorithm 2 Surveillance for Unknown Environments

Input: $\mathbf{P}$, $\mathbf{x}_p^0$, $\mathbf{x}_e^0$
1: $\hat{\mathbf{X}}_p \leftarrow \emptyset$; $\hat{\mathbf{X}}_e \leftarrow \emptyset$; $\mathcal{T} \leftarrow \emptyset$;
2: $\hat{\mathbf{P}} \leftarrow$ getWorkSpace($V(\mathbf{x}_p^0), V(\mathbf{x}_e^0)$);
3: $\hat{\mathbb{V}} \leftarrow$ getSubset($\hat{\mathbf{P}}, \{v_i\}$);
4: **for all** $v_i \in \mathbb{V}$ not in case (a) **do**
5: $\quad \hat{P}_i \leftarrow$ getApproxPreventionRegion($\hat{\mathbf{P}}, v_i$);
6: $\quad RRT_p^* \leftarrow$ getRRT*($\mathbf{x}_p^0, \hat{P}_i, \hat{\mathbf{P}}$);
7: $\quad \hat{\mathbf{x}}_p \leftarrow$ getBestTra(RRT_p^*);
8: $\quad \hat{\mathbf{X}}_p \leftarrow \hat{\mathbf{X}}_p \cup \hat{\mathbf{x}}_p$;
9: $\quad \hat{l}_i(0) \leftarrow$ getApproxFreeEdge($\hat{\mathbf{P}}, v_i$);
10: $\quad \hat{E}_i(0) \leftarrow$ getApproxEscapeRegion($\hat{\mathbf{P}}, \hat{l}_i(0)$);
11: $\quad RRT_e^* \leftarrow$ getRRT*($\mathbf{x}_e^0, \hat{E}_i(0), \hat{\mathbf{P}}$);
12: $\quad \hat{\mathbf{x}}_e \leftarrow$ getBestTra(RRT_e^*);
13: $\quad \hat{\mathbf{X}}_e \leftarrow \hat{\mathbf{X}}_e \cup \hat{\mathbf{x}}_e$;
14: $\quad \mathcal{T} \leftarrow \mathcal{T} \cup (\tau(\hat{P}_i, \hat{\mathbf{x}}_p) - \tau(\hat{E}_i(0), \hat{\mathbf{x}}_e))$;
15: **end for**
16: $i^* \leftarrow$ getArgMax($\mathcal{T}$);
17: $\hat{\mathbf{x}}_p^* \leftarrow \hat{\mathbf{X}}_p[i^*]$, $\hat{\mathbf{x}}_e^* \leftarrow \hat{\mathbf{X}}_e[i^*]$;
18: **return** $\hat{\mathbf{x}}_p^*$ and $\hat{\mathbf{x}}_e^*$;

4.4 Sensor with Limited Range

In the following, we consider that the environment is known to both players. However, the pursuer's visibility region is limited by its sensor range. Thus, two cases arise: (1) The evader decides to escape by reaching the boundary of the pursuer's visibility region without visiting a reflex vertex, in which case the pursuer tries to push the boundary away from the evader (refer to Fig. 9). (2) The evader tries to escape by reaching a reflex vertex $v_i \in \mathbb{V}$ in the pursuer's visibility region, in which case, the pursuer tries to reach the prevention region associated with that reflex vertex.

Algorithm 4 depicts the methodology to address escapes due to the limited sensor range. In Lines 2 and 3, two trees with k nodes are built from the initial player's states ($\mathbf{x}_p^0$, $\mathbf{x}_e^0$). Recalling that the computed strategies will be executed during a period of T_ϵ, in Line 4, a set $\mathbf{N}$ is formed with all the nodes $n_j \in RRT_p^*$ s.t. $\tau(n_j, \hat{\mathbf{x}}_p) \leq T_\epsilon$. Set $\mathbf{N}$ is a set of candidate goal sates for the pursuer (reachable within T_ϵ time units) where it can move to prevent an escape due to the limited sensor range; Lines 5 to 11 iterate on this set. For each pursuer state/node $n_j \in \mathbf{N}$, an escape region $\tilde{E}_j$ is computed as $\tilde{E}_j = \overline{\mathbf{P}^\circ \setminus B(n_j, \iota)}$, where ι is the sensor range, and $B(n_j, \iota)$ is a ball centred at n_j and radius ι (Line 7). Region $\tilde{E}_j$ models the portion of the environment beyond the range of the pursuer's sensor while standing at n_j. The minimum time trajectories for the players to reach their goals are computed in Lines 6 and 8 (for the purser trajectories to reach nodes n_j, and for the evader trajectories to reach regions $\tilde{E}_j$). The index j^* of the pursuer's best node to prevent an escape is retrieved in Line 12, and in Line 13, the respective best trajectories for the players are also retrieved.

In Line 2 of Algorithm 3, Algorithm 4 is called as a subroutine. The latter delivers the best strategy for the players to deal with the sensor range; its results are stored in Line 3. From Lines 5 to 15, Algorithm 3 deals with the escapes through reflex vertices in a similar manner as it was done Section 4.3. The main difference is that, since the environment is known to the play-

ers, the prevention and escape regions, P_i and $E_i(0)$, are computable. Finally, Lines 16 and 17 retrieve the worst case for the pursuer, considering both the analyses of potential escapes due to sensor range (Line 2) and through reflex vertices (Lines 5 to 15).

Algorithm 3 Surveillance with Limited Range

Input: $\mathbf{P}$, $\mathbf{x}_p^0$, $\mathbf{x}_e^0$
1: $\hat{\mathbf{X}}_p \leftarrow \emptyset$; $\hat{\mathbf{X}}_e \leftarrow \emptyset$; $\mathcal{T} \leftarrow \emptyset$;
2: $(\hat{\mathbf{x}}_p, \hat{\mathbf{x}}_e, \Psi^{(0)}) \leftarrow solveForRange(\mathbf{P}, \mathbf{x}_p^0, \mathbf{x}_e^0)$
3: $\hat{\mathbf{X}}_p \leftarrow \hat{\mathbf{X}}_p \cup \hat{\mathbf{x}}_p$, $\hat{\mathbf{X}}_e \leftarrow \hat{\mathbf{X}}_e \cup \hat{\mathbf{x}}_e$, $\mathcal{T} \leftarrow \mathcal{T} \cup \Psi^{(0)}$;
4: $\mathbb{V} \leftarrow \text{getSubset}(V(\mathbf{x}_p^0), \{v_i\})$;
5: **for all** $v_i \in \mathbb{V}$ **do**
6: $P_i \leftarrow \text{getPreventionRegion}(\mathbf{P}, v_i)$;
7: $RRT_p^* \leftarrow \text{getRRT}^*(\mathbf{x}_p^0, P_i, \mathbf{P})$;
8: $\hat{\mathbf{x}}_p \leftarrow \text{getBestTra}(RRT_p^*, P_i)$;
9: $l_i(0) \leftarrow \text{getFreeEdge}(\mathbf{P}, \hat{\mathbf{x}}_p(0), v_i)$;
10: $E_i(0) \leftarrow \text{getEscapeRegion}(\mathbf{P}, V(\hat{\mathbf{x}}_p(0)), l_i(0))$;
11: $RRT_e^* \leftarrow \text{getRRT}^*(\mathbf{x}_e^0, E_i(0), \mathbf{P})$;
12: $\hat{\mathbf{x}}_e \leftarrow \text{getBestTra}(RRT_e^*, E_i(0))$;
13: $\hat{\mathbf{X}}_p \leftarrow \hat{\mathbf{X}}_p \cup \hat{\mathbf{x}}_p$, $\hat{\mathbf{X}}_e \leftarrow \hat{\mathbf{X}}_e \cup \hat{\mathbf{x}}_e$;
14: $\mathcal{T} \leftarrow \mathcal{T} \cup (\tau(P_i, \hat{\mathbf{x}}_p) - \tau(E_i(0), \hat{\mathbf{x}}_e))$;
15: **end for**
16: $i^* \leftarrow \text{getArgMax}(\mathcal{T})$;
17: $\hat{\mathbf{x}}_p^* \leftarrow \hat{\mathbf{X}}_p[i^*]$, $\hat{\mathbf{x}}_e^* \leftarrow \hat{\mathbf{X}}_e[i^*]$;
18: **return** $\hat{\mathbf{x}}_p^*$ and $\hat{\mathbf{x}}_e^*$;

Algorithm 4 solveForRange(.)

Input: $\mathbf{P}$, $\mathbf{x}_p^0$, $\mathbf{x}_e^0$, ι
1: $\hat{\mathbf{X}}_p \leftarrow \emptyset$; $\hat{\mathbf{X}}_e \leftarrow \emptyset$; $\mathcal{T} \leftarrow \emptyset$;
2: $RRT_p^* \leftarrow \text{getRRT}^*(\mathbf{x}_p^0, k, \mathbf{P})$;
3: $RRT_e^* \leftarrow \text{getRRT}^*(\mathbf{x}_e^0, k, \mathbf{P})$;
4: $\mathbf{N} \leftarrow \text{getNodesSet}(RRT_p^*, T_\epsilon)$
5: **for all** $n_j \in \mathbf{N}$ **do**
6: $\hat{\mathbf{x}}_p \leftarrow \text{getBestTra}(RRT_p^*, n_j)$;
7: $\tilde{E}_j \leftarrow \text{getEscapeRegion}(\mathbf{P}, B(n_j, \iota))$;
8: $\hat{\mathbf{x}}_e \leftarrow \text{getBestTra}(RRT_e^*, \tilde{E}_j)$;
9: $\hat{\mathbf{X}}_p \leftarrow \hat{\mathbf{X}}_p \cup \hat{\mathbf{x}}_p$, $\hat{\mathbf{X}}_e \leftarrow \hat{\mathbf{X}}_e \cup \hat{\mathbf{x}}_e$;
10: $\mathcal{T} \leftarrow \mathcal{T} \cup (\tau(n_j, \hat{\mathbf{x}}_p) - \tau(\tilde{E}_j, \hat{\mathbf{x}}_e))$;
11: **end for**
12: $j^* \leftarrow \text{getArgMin}(\mathcal{T})$;
13: $\hat{\mathbf{x}}_p^* \leftarrow \hat{\mathbf{X}}_p[j^*]$, $\hat{\mathbf{x}}_e^* \leftarrow \hat{\mathbf{X}}_e[j^*]$;
14: **return** $\hat{\mathbf{x}}_p^*$ and $\hat{\mathbf{x}}_e^*$ and $\mathcal{T}[j^*]$;

5 Description of the Surveillance Algorithms through the Visibility-based Formulation

Problem 1 provides a general formulation in which the surveillance algorithms presented in Section 4 correspond to specializations of that formulation; in this section, we elaborate on how those specializations are connected to the general formulation. To describe the different specializations, we will make use of the next simplification of Equation (2)

$$\Psi = \max_{v_i \in \Omega} \{\Psi_i\}. \tag{3}$$

Term Ψ_i refers to the time difference between the duration of the pursuer's and evader's executions of their optimal behaviors; that term is defined according to the different addressed scenarios. Additionally, the set Ω defines the search space of the worst-case vertex for the pursuer.

For the scenario of the visibility-based game with the known environment and unlimited sensor range, Ψ_i is defined as follows

$$\Psi_i \stackrel{\text{def}}{=} \max_{t_j \in \{t_0, t_1, ..., t_M\}} \{t_j - \min_{\hat{\mathbf{x}}_e} \{\tau(E_i(t_j), \hat{\mathbf{x}}_e)\}\}. \tag{4}$$

Equation (4) considers that, given start poses $\mathbf{x}_e^0$ and $\mathbf{x}_p^0$, trees RRT_p^* and RRT_e^* have already been computed. Functions $\hat{\mathbf{x}}_p$ and $\hat{\mathbf{x}}_e$ refer to trajectories encoded in RRT_p^* and RRT_e^*, respectively. Firstly, a trajectory $\hat{\mathbf{x}}_p^* = \arg\min_{\hat{\mathbf{x}}_p} \{\tau(P_i, \hat{\mathbf{x}}_p)\}$ is computed, and its duration is divided into M intervals to retrieve the set $\{t_0, t_1, ..., t_M\}$. The escape region $E_i(t)$ is assumed to evolve according to $\hat{\mathbf{x}}_p^*$. The vertex search set $\Omega \stackrel{\text{def}}{=} \mathbb{V}$.

For the scenario of unknown environment, Ψ_i is defined as follows

$$\Psi_i \stackrel{\text{def}}{=} \min_{\hat{\mathbf{x}}_p} \{\tau(\hat{P}_i, \hat{\mathbf{x}}_p)\} - \min_{\hat{\mathbf{x}}_e} \{\tau(\hat{E}_i(0), \hat{\mathbf{x}}_e)\}, \tag{5}$$

where $\hat{\mathbf{x}}_p$ and $\hat{\mathbf{x}}_e$ again refer to trajectories encoded in the respective trees. Regions $\hat{P}_i$ and $\hat{E}_i(0)$ are computed as described in Section 4.3. Set $\Omega \stackrel{\text{def}}{=} \hat{\mathbb{V}}$.

For the scenario of a known environment considering a sensor with limited range, the set $\Omega \stackrel{\text{def}}{=} \mathbb{V} \cup v_0$, where the virtual vertex v_0 is used to represent the case when the escape is due to the sensor limit range. Thus, Ψ_i is defined as follows

$$\Psi_i \stackrel{\text{def}}{=} \begin{cases} \Psi^{(0)} & i = 0, \quad \text{(limited range)} \\ \Psi^{(i)} & \text{otherwise.} \quad \text{(vertex)} \end{cases} \tag{6}$$

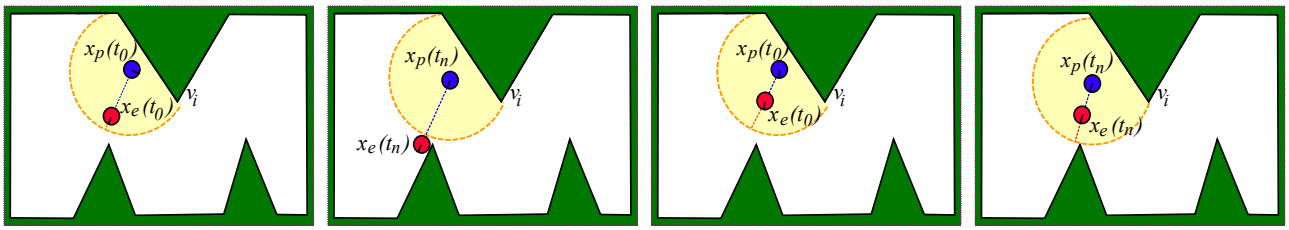
The case of escape due to limited range yields the next expression

$$\Psi^{(0)} \stackrel{\text{def}}{=} \min_{n_j \in \mathbf{N}} \{\tau(n_j, \hat{\mathbf{x}}_p) - \min_{\hat{\mathbf{x}}_e} \{\tau(\tilde{E}_j, \hat{\mathbf{x}}_e)\}\}. \tag{7}$$

Note that $\tau(n_j, \hat{\mathbf{x}}_p)$ is unique since n_j and $\hat{\mathbf{x}}_p$ belong to the tree RRT_p^*. The set $\mathbf{N}$ and region $\tilde{E}_j$ are defined as in Section 4.4. The case of escape using a vertex yields the next expression

$$\Psi^{(i)} \stackrel{\text{def}}{=} \min_{\hat{\mathbf{x}}_p} \{\tau(P_i, \hat{\mathbf{x}}_p)\} - \min_{\hat{\mathbf{x}}_e} \{\tau(E_i(0), \hat{\mathbf{x}}_e)\}. \tag{8}$$

That concludes the description of the connections of section's 4 algorithms for the three addressed scenarios to the general formulation provided in Section 2.2.



(a) (b) (c) (d)

Fig. 9: In (a), the evader tries to reach the boundary of the pursuer's visibility region (yellow disc) without visiting a reflex vertex. In this case, the pursuer cannot push the boundary away from the evader before that player reaches it, as seen in (b). In (c), again, the evader tries to escape by reaching the boundary of the pursuer's visibility region; however, in this case, the pursuer's motion allows him to maintain visibility of the evader by pushing the boundary away from it, as shown in (d). The orange dashed line represents the evader's minimum time trajectory if it tries to reach the boundary. The blue dashed line represents the pursuer's trajectory to push the boundary away from the evader.

6 Simulation Experiments

We start by presenting simulations in a scenario where the whole environment is known to both players, followed by simulations where the players compute their strategies merely using their local representations of the environment. For both scenarios, we assume an unlimited range sensor. After those simulations, we present the case where we consider a known environment and a sensor with a limited range. For the sake of completeness, we also show a simulation with an unknown environment with a limited range sensor, which can be directly obtained based on the concepts elaborated for the other scenarios. In all the previous experiments, the players execute their optimal strategies. We also include a simulation with an evader that acts irrationally. In that simulation, for brevity, we present the scenario of a known environment and unlimited sensor range; however, irrational players can also be considered in the other two scenarios. Thereafter, to further show the flexibility of our approach, we present a simulation trial where the RRT* is substituted with a SST* and the addressed system is replaced with a DDR with second-order dynamics. Subsequently, we present a simulation considering a sequence of escape and prevention regions, and a visualization in 3D. Finally, an evaluation is presented in which we study the impact of a couple of parameters in the proposed approach; namely, the parameter M used to discretize the pursuer's trajectory and the number of samples in the RRT*. The computations were done in an Intel Core i7-7700HQ @ 2.80 GHz, 16.0 GB RAM, Windows 10. The algorithm was programmed in C# and C++, and the results were plotted with matplotlib (python).

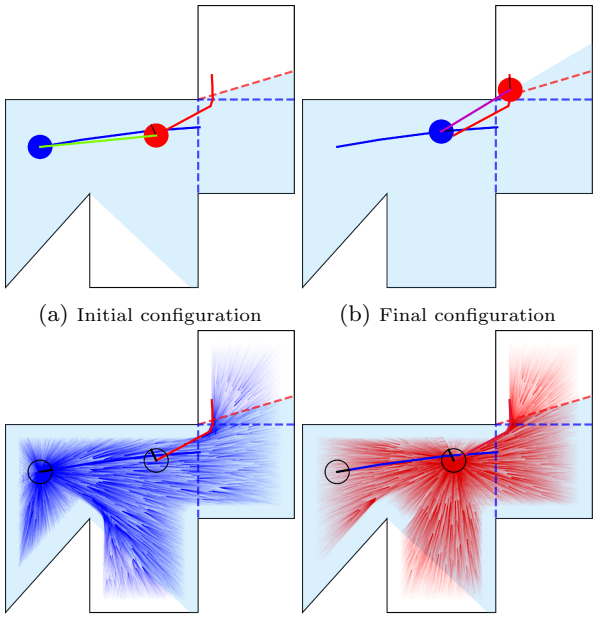
6.1 Known environment with unlimited range sensor

Fig. 10 shows a simulation experiment for a scenario where the global map is known to the players. In that simulation, the evader escapes the pursuer's surveillance breaking visibility behind a corner. The pursuer is represented with a blue disc, and its trajectory is shown in blue. The evader is depicted as a red disc, and its trajectory is shown in red. The players start at the leftmost configurations of their respective trajectories (see Fig. 10a). The pursuer's visibility region $V(\mathbf{x}_p(t))$ is shown in light blue. The initial state of the free edge chosen by the evader to escape, $l_i(0)$, is shown with a red dashed segment. The green line segment between the two players represents the line of sight between them. The configurations of the players after the evader breaks surveillance are shown in Fig. 10b. Fig. 10c shows in blue the pursuer's tree of trajectories computed with an RRT* using as steering procedure the planner presented in [2]. The evader's tree of trajectories is shown in Fig 10d. The shown trees correspond to their projections onto the plane.

Fig. 11 shows a simulation experiment where the pursuer can maintain the evader's visibility when the global map is available to both players. Fig. 11a shows the initial poses of the players with the free edge (red dashed segment), $l_i(0)$, to which the evader aims to escape at the beginning of the game. Fig. 11b shows the moment when the pursuer reaches the prevention region, which happens before the evader reaches the corresponding free edge. The line of sight between the players is shown in green.

6.2 Unknown environment with unlimited range sensor

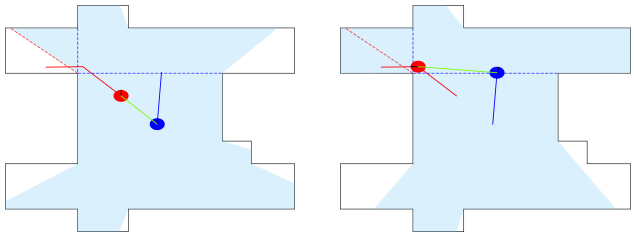
Fig. 12 shows an experiment in which the environments' global map is unknown for the players. The strategies



(a) Initial configuration (b) Final configuration

(c) Pursuer's tree. (d) Evader's tree.

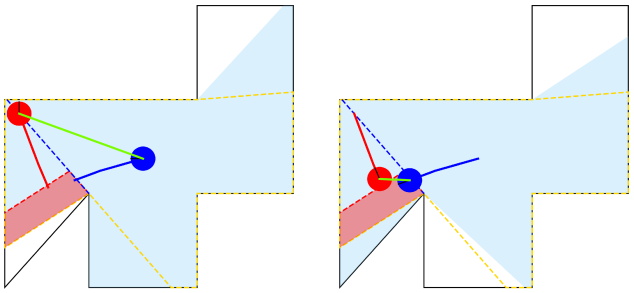
Fig. 10: Known environment scenario. The evader eventually breaks visibility with the pursuer at the configuration shown in Fig. 10b.



(a) Initial configuration. (b) Final configuration.

Fig. 11: Known environment scenario. The pursuer is able to reach the prevention region before the evader breaks visibility with it.

of the players are based on the local map observed by the players' sensors. The depicted scenario corresponds to an instance of the case (d) in Subsection 4.3. Fig. 12a shows the initial poses of the players. The intersection of both visibility regions, $V(\mathbf{x}_p(t)) \cap V(\mathbf{x}_e(t))$, is delimited by a yellow dashed contour. The prevention region's border that the pursuer can compute with the information from the local map observed with its sensor is shown with a blue segment. The evader's region taken as a goal for the RRT* algorithm corresponds to the red band. That region is adjacent to the vertex generating the free edge that the evader will use to try to escape. The planned evader's path is shown in red, and the pursuer's path is shown in blue. In this simulation experiment, the pursuer can keep the evader's surveillance. Fig. 12b shows the moment when the pursuer reaches its goal region.

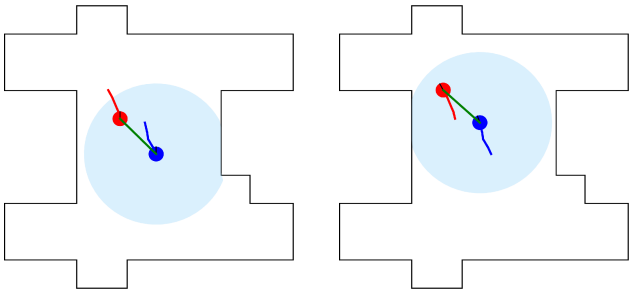


(a) Initial configuration. (b) Final configuration.

Fig. 12: Unknown environment scenario corresponding to an instance of case (d) in Subsection 4.3. The pursuer can reach the prevention region keeping surveillance of the evader.

6.3 Known environment with limited range sensor

Fig. 13 shows a simulation where the pursuer has a visibility region limited by range. The light blue circle represents the pursuer's visibility region. The initial configurations of the players are depicted in Fig. 13a. In this case, the evader tries to escape by reaching the boundary of the region. The pursuer keeps surveillance following a motion strategy that pushes the boundary away from the evader. Fig. 13b shows the configuration of the players at the end of the experiment. The evader is still inside of the pursuer's visibility region.



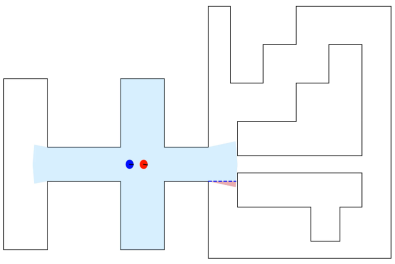
(a) Initial configuration. (b) Final configuration.

Fig. 13: The pursuer has a visibility region limited by range (light blue circle). The evader tries to escape by reaching the boundary of the region. However, the pursuer keeps surveillance following a motion strategy that pushes the boundary away from the evader.

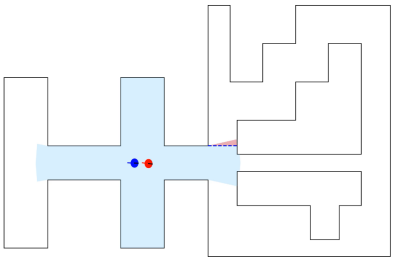
6.4 Unknown environment with limited range sensor

The present case is similar to the one of a sensor with limited range in a known environment, only with the following modifications: instead of utilizing regions P_i and $E_i(0)$ for the pursuer and the evader, respectively, regions $\hat{P}_i$ and $\hat{E}_i(0)$ are employed, which are computed as described in Section 4.3.

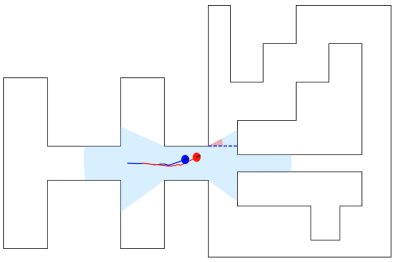
Fig. 14 shows a simulation of this case. The initial configurations for the players are presented in Fig. 14a. Two intermediate configurations are show in Fig. 14b and Fig. 14c. The evader breaks surveillance by reaching the escape region (see. Fig. 14d).



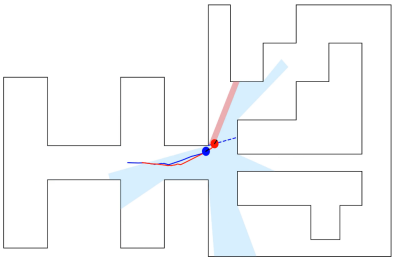
(a) Initial configuration.



(b) Intermediate configuration.



(c) Intermediate configuration.

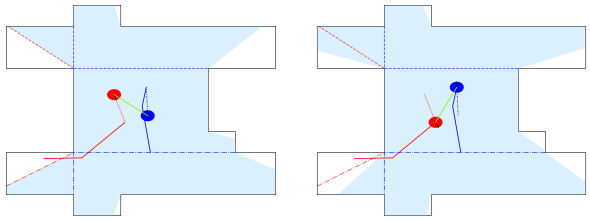


(d) Final configuration.

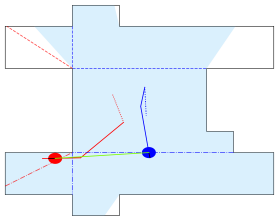
Fig. 14: An unknown environment considering a pursuer with limited sensor range. The evader breaks surveillance by reaching the escape region.

6.5 Irrational players in known environment

Fig. 15 shows a simulation where the evader initially does not follow its optimal strategy in a known environment. The initial configurations of the players are portrayed in Fig. 15a. For those configurations, the evader's optimal strategy corresponds to reach the free edge (red dashed segment) located in the upper left part of the environment. At the same time, the pursuer needs to reach the associated prevention region to maintain surveillance. However, instead of following its optimal strategy, the evader travels the trajectory depicted as the red dotted segment moving away from that free edge. The pursuer, on the other side, follows its optimal strategy. After some time (see Fig. 15b), the evader reaches a configuration where the optimal strategies of both players change. From that moment, the evader also starts to follow its optimal strategy. Note that for the players' configurations shown in Fig. 15b, the evader's optimal strategy is now reaching the free edge at the bottom left part of the environment; the pursuer must reach the associated prevention region to keep surveillance. Despite the evader not following its optimal strategy at the beginning of the game and its motion leads it to a configuration where both players' optimal strategies are different from the initial ones, the pursuer can react on time. It prevents an escape when the evader starts to follow its optimal strategy (see Fig. 15c).



(a) (b)



(c)

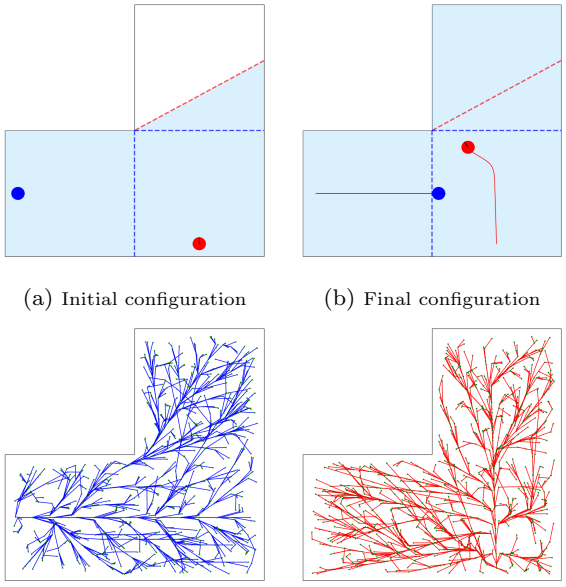
Fig. 15: An irrational evader in a known environment. (a) The evader travels the red dotted line trajectory (suboptimal) while the pursuer follows its optimal trajectory (blue dotted line). (b) After some time, the evader reaches a configuration where both players' motion strategies change; for that moment, it also starts to follow its optimal motion trajectory (red segments). (c) Although the evader does not follow the optimal strategy at the beginning of the game, and its motion leads him to a configuration where the optimal strategies of both players change, the pursuer can react on time and maintain the evader's visibility.

6.6 Alternative sampling-based algorithms

We have included a simulation experiment using a SST* [25] (see Fig. 16). That allowed us to consider second-order dynamics for a DDR since the SST* does not need to solve a two-point boundary value problem (BVP). In particular, the DDR is controlled through the acceleration of the right and left wheels [32]. The purpose of this simulation is to show that our approach is general enough to work with other alternatives to the RRT* and can handle other types of mobile robots.

Fig. 16a shows the initial configuration of the players. We consider the case of a known environment and an unbounded sensing range for the pursuer. The final configurations and the trajectories traveled by the players are shown in Fig. 16b. In this case, the pursuer can maintain surveillance of the pursuer. One can observe that the evader's trajectory is smooth given the second-order dynamics of the model.

Figs. 16c and 16d present the resulting trees for each player, where the sparse nature of the SST* can be appreciated. The computation time of the trees' construction was in the order of a couple of minutes. Initially, the algorithm considered 25,000 samples in the inner SST, and the SST* was run for 10 iterations, where the last iteration generated around 1 million samples for the inner SST. The final pursuer tree has 989 nodes, while the evader tree has 1,093 nodes. Therefore, the SST* construction took significantly more time compared with what we observed in the previous RRT* experiments; however, keep in mind that the addressed system is different in this simulation. On the other hand, since the SST* is sparse, its queries take less time than those of the RRT*. An alternative to improve the tree construction time might be to only use a SST without the SST* framework, but that would incur the expense of getting asymptotically near-optimality instead.



(a) Initial configuration (b) Final configuration

(c) Pursuer's tree. (d) Evader's tree.

Fig. 16: Simulation considering the SST* algorithm to compute the players' strategies. The environment is known, and the pursuer has an unbounded sensing range. The pursuer maintains surveillance of the evader.

6.7 A simulation in a 3D environment considering a sequence of escape and prevention regions

Fig. 17 shows a simulation in a 3D environment. The pursuer is equipped with an unlimited range sensor, and the RRT* is again utilized to generate the trajectories considering the kinematic model from Section 2.1. The environment map is known for both players, and they move in a 3D representation of a museum. The blue and red lines in Fig. 17 show the trajectories of the pursuer and the evader, respectively. Those trajectories are computed sequentially, considering only a single visit in the future to a pair of regions at a time. The planning stage is executed in a 2D representation (plane) of the map. Such representation could be obtained by projecting the convex hulls of the environment's objects onto the plane; note that planning with that projection is a conservative approach for keeping track of the evader. This simulation shows that it is possible to keep track of the evader in the long term by planning sequentially for one pair of escape and prevention regions at a time. However, keep in mind that planning in that decoupled way, there is no theoretical guarantee that the pursuer will keep the evader's surveillance in the long run. Fig. 18 shows several screenshots of the trajectories traveled by the players. In each row of that figure, we show two views of the same player's position. The ones on the left side correspond to a camera located outside of the robots and on the right to a camera located on the pursuer.

In the *multimedia material*, we have included a video ("video.mp4") showing the simulation of six experiments similar to the ones above. It is worth mentioning that in the previous simulations, the players' velocities are assumed equal. However, note that a given RRT* computes the optimal trajectories from a given configuration to several regions in the environment. The underlying structure of the resulting tree remains fundamentally the same regardless of the player's velocities. Hence, it is straightforward to change the velocity parameter to travel the trajectories faster or slower.

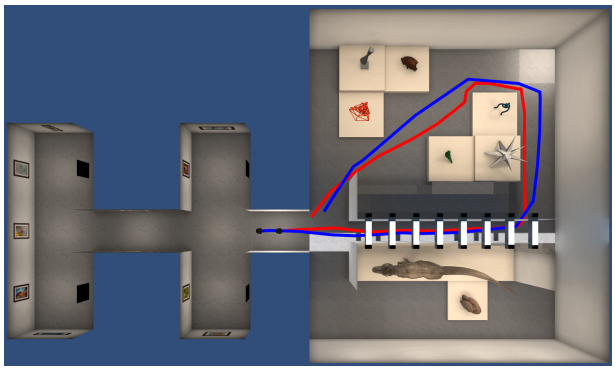


Fig. 17: In this simulation, the players travel in a 3D representation of a museum. They know the map of the environment. The pursuer is equipped with an unlimited range sensor. The trajectories of the pursuer and the evader correspond to the blue and red lines, respectively. During the route, the players have to solve a set of different local planning problems. Even though a global plan is not performed, the pursuer can maintain the evader's visibility for more than one pair of escape and prevention regions in this case.

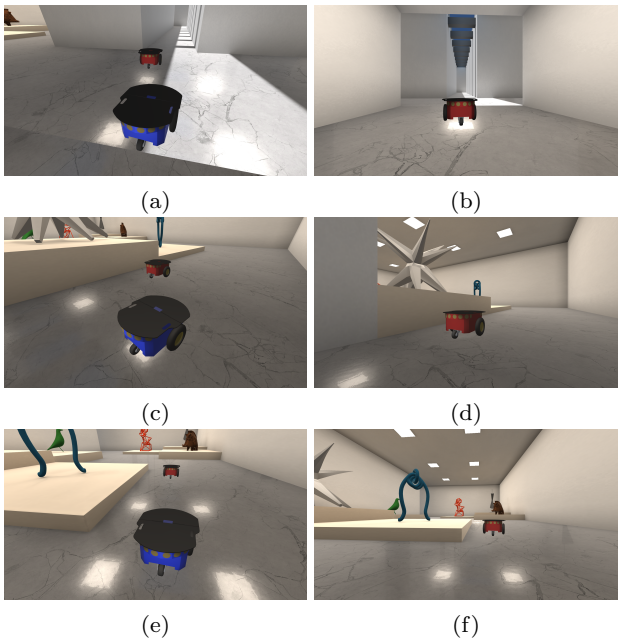


Fig. 18: Screenshots along the trajectories followed by the players in Fig. 17. The ones on the left side correspond to a camera located outside of the robots and on the right to a camera located on the pursuer.

As final note, the planning stage takes from two to ten seconds in our implementation, depending on the number of nodes in the RRT*. A more reactive strategy could be generated if a faster method to compute the shortest time trajectories would be available.

To hasten our software's execution, the present work might use existing research focused on improving the computation time of the RRT*, either using efficient data structures [14] or employing specialized hardware [35]. Moreover, recent works have shown that it is possible to hasten the execution of sample-based algorithms employing learning-based approaches [29]. Nonetheless, this is a research topic by itself that we leave for future work.

6.8 Parameters evaluation

In the present section, we evaluate the impact of different parameters on the proposed formulation. In particular, we assess the effect of parameter M used to discretize the trajectory of the pursuer so that the evader considers M escape regions to determine the shortest escape path. Subsequently, we explore the impact of reducing the number of samples for the RRT*, thus getting farther from the optimal solution, on the ability to keep surveillance.

Concerning parameter M, we have studied two different environments, one with a single corner and another with holes, and three initial configurations (see Fig.19), yielding three different settings. Without loss of generality, we focus on the known environment scenario with unlimited sensor range. The analysis consists of considering an initial configuration and compute the value of Ψ for different values of M ranging from 10 to 200. Recall that a value of $\Psi \leq 0$ translates into the pursuer being able to prevent an evader's escape. The results are shown in Fig. 20 for each of the individual settings from Fig. 19. Additionally, in each chart, three plots are shown corresponding to different bounds on the players' controls, $V^{\max}$.

From Fig. 20, it can be observed that the computed value of Ψ significantly varies with respect to M. However, it tends to converge to a certain value as M increases, which is especially observed in Figs. 20a and 20b. Different trace behaviors were observed for the three settings. Interestingly, even though both players had the same control bound $V^{\max}$ during a play, this speed influenced the computed value of Ψ. Also, the form of the trace of Ψ significantly changed even when the same environment was used, and only the initial pose of the pursuer was changed, see Figs. 19b and 19c, and Figs. 20b and 20c, respectively. Lastly, for the case of Fig. 20a (Fig. 19a), even when the value of M increased, the value of Ψ remained below zero, meaning that no escape path for the evader has been found. Similar behavior was seen in the case of Fig. 20b (Fig. 19b) for certain $V^{\max}$ values; nonetheless, for $V^{\max} = 1.0$, the computed Ψ was above zero even for small values of M, meaning that escape paths for the evader were already found. Conversely, in Fig. 20c (Fig. 19c) there is a transition of Ψ from negative to positive values. Consequently, as expected, reducing the parameter M

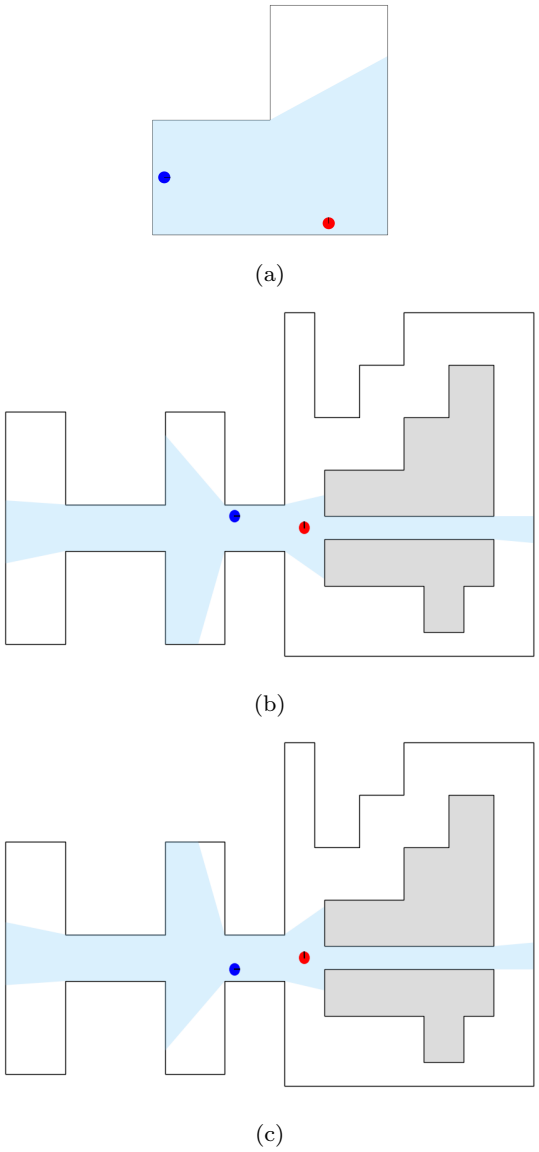


Fig. 19: The two different environments and three initial configurations considered for evaluating the impact of parameter M.

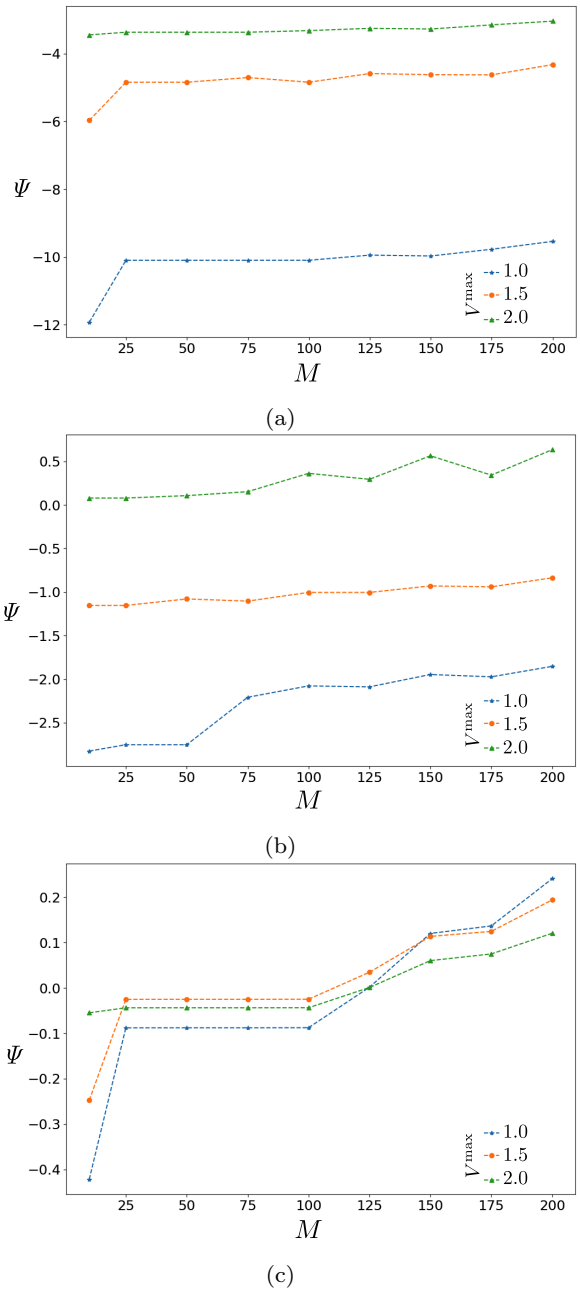


Fig. 20: The plots present the evolution of the computed value of Ψ as M increases. Each figure corresponds to a respective setting (the one with the same label) in Fig. 19. Three different values of $V^{\max}$ were explored. The same $V^{\max}$ value was employed for both players in each play.

reduces the approach's capability to identify properly potential escape opportunities for the evader. The M value has to be large enough to identify the evader's potential escapes but low enough to not significantly affect the computational cost of the proposed approach.

Regarding studying the impact of the number of samples in the RRT* on the ability to keep surveillance, we employed the same three settings depicted in Fig. 19. Fig. 21 shows the evolution of the value of Ψ as the number of nodes in the RRT* increases by the same amount for both players. The analysis considers an initial configuration and computes the value of Ψ for different numbers of nodes in the RRT*, ranging from 10000 to 100000. The shown plots correspond to the execution of three trials, one per setting in Fig. 19.

It is worth noticing that the plots do not show a monotonic behavior, but we can observe that as the number of nodes increases, Ψ stabilizes and seems to converge to a given value. Moreover, depending on the plot, we observe an increasing or decreasing trend on Ψ as the number of samples increase. The increasing trend

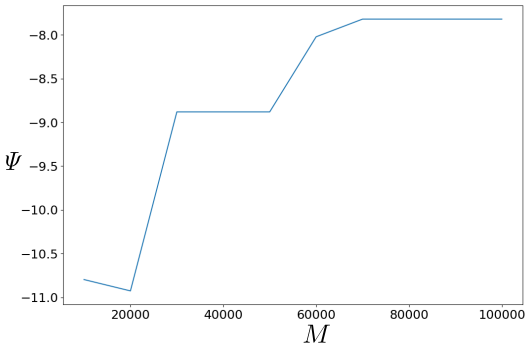
might be due to the improvement of the quality of the evader's path at a higher rate than that of the pursuer's path, and in the decreasing trend, the pursuer's path might be improving faster than the evader's path.

As expected, based on the observed convergence in the Ψ value, as the samples within the RRT* increase, the approach enhances its capability of determining if the evader has or not a winning strategy. That is to say, considering more samples within the RRT* will enhance the ability of the approach to discover better quality paths for the evader, allowing the pursuer to react accordingly and consequently improve the surveillance capabilities. In addition, the quality of the pursuer's path is also improved, and thus the surveillance capabilities are improved.

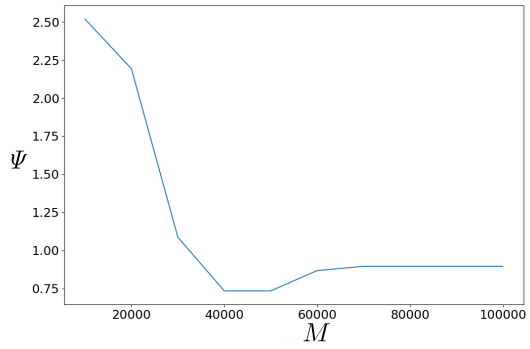
7 Conclusions and Future Work

Many works have addressed visibility-based pursuit-evasion games; however, that problem has not been completely solved yet. The works found in the literature consider different simplifications, e.g., point robots, holonomic robots, a single corner analysis, among others. Note that in this paper, a more complex problem was addressed in which the players are nonholonomic disc-shaped robots in an environment with several obstacles that represent multiple possibilities to escape. Further complicating matters is that the pursuer's visibility region changes as the pursuer moves, and consequently, its borders corresponding to the evader's goals also change. Hence, a set of planning problems must be solved for the evader. In this paper, we have proposed a method able to do that.

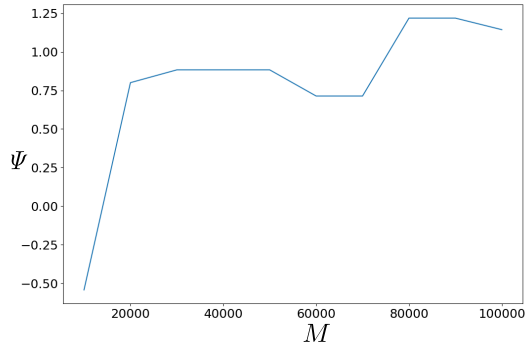
This work's proposed approach was based on comparing the shortest-time trajectories' duration of the players to reach their goals, which results in a worst-case analysis for the pursuer. Under such formulation, a sufficient condition for the pursuer to maintain the evader's visibility was also established. An RRT* approach with a local planner based on optimal control (steering procedure) was used to determine the minimum time trajectories for the nonholonomic players. The idea of approximating the players' optimal paths with an RRT*, either to escape (for the evader) or to prevent the escaping (for the pursuer), has already been used before for other pursuit-evasion games, for instance, in [17]. Nonetheless, we stress that the formulation proposed in this work is novel to classical differential game modeling, where there is no clear termination condition when the pursuer can maintain evader visibility forever. Even more, the formulation is flexible enough to be able to address different aspects of our problem.



(a)



(b)



(c)

Fig. 21: The plots present the evolution of the computed value of Ψ as the number of nodes increases. Each figure corresponds to a respective setting (the one with the same label) in Fig. 19.

Within the studied aspects of the addressed game is the player's knowledge of the environment. In that context, we have proposed an approach that does not require knowing the global map of the environment. The strategies are based on incomplete information obtained from a local map observed by the players' sensors; more precisely, they are computed using the intersection of the players' visibility regions, which was also

proven not to imply any cooperation between the players. Our approach also allows one to deal with optimal or suboptimal players who might fool the opponent by performing suboptimal actions. That happens because the players can compute their best strategies based only on their current pose and the map; one player's action or goal does not depend on the other player's action or goal. However, it is understood that there is a real-time issue; the approach can be implemented in closed-loop provided that the computation of the trees for the players can be done fast enough. Additionally, other aspects, such as the limited sensor range, were also considered, along with other sampling-based alternatives to the RRT*, enabling the method to consider other dynamical systems.

As future work, we would like to propose procedures for hastening the computation of the players' strategies yielding real-time motion strategies. We would also like to study in more detail the effect of generating short-term plans, which require shorter processing times and allow the players to react more often to deviations from the planned trajectories. We are also interested in extending our approach to consider a pursuer with a limited field of view. Another interesting research direction is computing motion strategies for a 3D environment, for example, a robot moving across a hill; in this case, it is possible to compute 3D visibility using sampling from the workspace [31]. Finally, we believe that it should be straightforward to extend our approach to deal with heterogeneous robots (holonomic vs. nonholonomic robots, robots with different controls or speeds); nonetheless, it is something worth exploring.

References

1. T. Başar and G. Olsder. *Dynamic Noncooperative Game Theory*. SIAM Series in Classics in Applied Mathematics, 1999.
2. D. J. Balkcom and M. T. Mason. Time optimal trajectories for bounded velocity differential drive vehicles. *The International Journal of Robotics Research*, 21(3):199–217, 2002.
3. I. Becerra, R. Murrieta-Cid, R. Monroy, S. Hutchinson, and J. P. Laumond. Maintaining strong mutual visibility of an evader moving over the reduced visibility graph. *Auton. Robots*, 40(2):395–423, 2016.
4. D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.
5. S. Bharadwaj, L. Ly, B. Wu, R. Tsai, and U. Topcu. Strategy synthesis for surveillance-evasion games with learning-enabled visibility optimization. In *IEEE Conference on Decision and Control (CDC)*, pages 6275–6281, 2019.
6. S. Bhattacharya and S. Hutchinson. On the existence of nash equilibrium for a two player pursuit-evasion game with visibility constraints. In *Algorithmic Foundation of Robotics VIII*, pages 251–265. Springer, 2009.
7. S. Bhattacharya and S. Hutchinson. A cell decomposition approach to visibility-based pursuit evasion among obstacles. *The International Journal of Robotics Research*, 30(14):1709–1727, 2011.
8. S. Bhattacharya, S. Hutchinson, and T. Basar. Game-theoretic analysis of a visibility based pursuit-evasion game in the presence of obstacles. In *2009 American Control Conference*, pages 373–378, June 2009.
9. T. H. Chung, G. A. Hollinger, and V. Isler. Search and pursuit-evasion in mobile robotics - A survey. *Auton. Robots*, 31(4):299–316, 2011.
10. A. Friedman. *Differential Games*. Dover Ed edition, 2006.
11. B. P. Gerkey, S. Thrun, and G. Gordon. Visibility-based pursuit-evasion with limited field of view. *The International Journal of Robotics Research*, 25(4):299–315, 2006.
12. L. J. Guibas, J. C. Latombe, S. M. LaValle, D. Lin, and R. Motwani. Visibility-based pursuit-evasion in a polygonal environment. In *Workshop on Algorithms and Data Structures*, pages 17–30. Springer, 1997.
13. L. J. Guibas, J. C. Latombe, S. M. LaValle, D. Lin, and R. Motwani. A visibility-based pursuit-evasion problem. *International Journal of Computational Geometry and Applications*, 9(4-5):471–493, 1999.
14. J. Ichnowski and R. Alterovitz. Concurrent nearest-neighbor searching for parallel sampling-based motion planning in so(3), se(3), and euclidean topologies. In *Algorithmic Foundations of Robotics XIII, WAFR 2018, Mérida México*, pages 69–85, 2018.
15. R. Isaacs. *Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization*. Courier Corporation, 1965.
16. V. Isler, C. Belta, K. Daniilidis, and G. J. Pappas. Hybrid control for visibility-based pursuit-evasion games. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, September 28 - October 2, 2004*, pages 1432–1437, 2004.
17. S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for a class of pursuit-evasion games. In *Algorithmic Foundations of Robotics IX - Selected Contributions of the Ninth International Workshop on the Algorithmic Foundations of Robotics, WAFR 2010, Singapore, December 13-15, 2010*, pages 71–87, 2010.
18. S. Karaman and E. Frazzoli. Optimal kinodynamic motion planning using incremental sampling-based methods. In *IEEE Conference on Decision and Control*, pages 7681–7687, 2010.
19. S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.
20. S. Karaman and E. Frazzoli. Sampling-based optimal motion planning for non-holonomic dynamical systems. In *2013 IEEE International Conference on Robotics and Automation*, pages 5041–5047, May 2013.
21. J. C. Latombe. Dynamic adaptation of individual perception-action control plans in a heterogeneous team of intelligent mobile agents. Technical report, STANFORD UNIV CA, 1997.
22. J. P. Laumond et al. *Robot motion planning and control*, volume 229. Springer, 1998.
23. S. M. LaValle, H. González-Banos, C. Becker, and J.C. Latombe. Motion strategies for maintaining visibility of a moving target. In *Proceedings of International Conference on Robotics and Automation*, volume 1, pages 731–736. IEEE, 1997.
24. A. Quattrini Li, R. Fioratto, F. Amigoni, and V. Isler. A search-based approach to solve pursuit-evasion games

with limited visibility in polygonal environments. In *Proceedings of the 17th International Conference on Autonomous Agents and Multi-Agent Systems*, pages 1693–1701, 2018.
25. Y. Li, Z. Littlefield, and K. E. Bekris. Asymptotically optimal sampling-based kinodynamic planning. *The International Journal of Robotics Research*, 35(5):528–564, 2016.
26. V. Macias, I. Becerra, R. Murrieta-Cid, H. M. Becerra, and S. Hutchinson. Image feedback based optimal control and the value of information in a differential game. *Automatica*, 90:271–285, 2018.
27. R. Murrieta-Cid, T. Muppirala, A. Sarmiento, S. Bhattacharya, and S. Hutchinson. Surveillance strategies for a pursuer with finite sensor range. *The International Journal of Robotics Research*, 26(3):233–253, 2007.
28. L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko. *The Mathematical Theory of Optimal Processes*. John Wiley, 1962.
29. A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip. Motion planning networks: Bridging the gap between learning-based and classical motion planners. *IEEE Transactions on Robotics*, 37(1):48–66, 2020.
30. U. Ruiz, R. Murrieta-Cid, and J. L. Marroquin. Time-optimal motion strategies for capturing an omnidirectional evader using a differential drive robot. *IEEE Transactions on Robotics*, 29(5):1180–1196, Oct 2013.
31. A. Sarmiento, R. Murrieta-Cid, and S. Hutchinson. A sample-based convex cover for rapidly finding an object in a 3-d environment. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3486–3491, 2005.
32. P. Soueres and J. D. Boissonnat. Optimal trajectories for nonholonomic mobile robots. In *Robot motion planning and control*, pages 93–170. Springer, 1998.
33. N. M. Stiffler and J. M. O'Kane. Planning for robust visibility-based pursuit-evasion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6641–6648. IEEE, 2020.
34. B. Tovar and S. M. LaValle. Visibility-based pursuit-evasion with bounded speed. *The International Journal of Robotics Research*, 27(11-12):1350–1360, 2008.
35. S. Xiao, N. Bergmann, and A. Postula. Parallel rrt* architecture design for motion planning. In *International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–4, 2017.
36. Z. Zhang, J. M. Smereka, J. Lee, L. Zhou, Y. Sung, and P. Tokekar. Game tree search for minimizing detectability and maximizing visibility. *Autonomous Robots*, 45(2):283–297, 2021.
37. J. Zhi, Y. Hao, C. Vo, M. Morales, and J. M. Lien. Computing 3-d from-region visibility using visibility integrity. *IEEE Robotics and Automation Letters*, 4(4):4286–4291, 2019.
38. R. Zou and S. Bhattacharya. Approximation of capture sets in visibility-based target-tracking games for nonholonomic players. In *Dynamic Systems and Control Conference*, volume 58288, page V002T07A004. American Society of Mechanical Engineers, 2017.
39. R. Zou and S. Bhattacharya. On optimal pursuit trajectories for visibility-based target-tracking game. *IEEE Trans. Robotics*, 35(2):449–465, 2019.

Eliezer Lozano is currently (2021) a second-year PhD student in computer science at the Centro de Investigacion en Matematicas, CIMAT, Guanajuato, México. He received a bachelor's degree in Physics and Mathematics from the Universidad Michoacana de San Nicolas de Hidalgo, Morelia, Mexico, in 2017 and a master's degree in physical engineering from the Universidad Michoacana de San Nicolas de Hidalgo, Morelia, Mexico, in 2019. His research interests include robotics and robot motion planning.

Israel Becerra received a B.S. degree in Mechatronics Engineering from the Monterrey Institute of Technology and Higher Education, CEM (2007). He received the M.Sc. degree (2010) and Ph.D. degree (2015), both in Computer Science, from Centro de Investigacion en Matematicas, CIMAT, Guanajuato, Mexico, where he also currently works as a researcher. He was a postdoctoral fellow in the Computer Science department of the University of Illinois at Urbana-Champaign from 2016 to 2017. He is mainly interested in motion planning, mobile robotics, pursuit-evasion games, optimal control, and virtual reality.

Ubaldo Ruiz received a Ph.D. degree in Computer Science from Centro de Investigacion en Matematicas, CIMAT, Guanajuato, Mexico, in 2013. In 2013-2014, he was a Postdoctoral Fellow in the Computer Science Department of the University of Minnesota. Since 2014, Ubaldo is a CONACYT Research Fellow working at Centro de Investigacion Cientifica y de Educacion Superior de Ensenada, CICESE, Ensenada, Mexico. His research interests are robotics, differential games, optimal control, and motion planning.

Luis Bravo received the B.S. in Computer Science from Universidad de Guanajuato, Mexico, in 2018 and the M.S. in Computer Science and Industrial Mathematics from Centro de Investigacion en Matematicas, Guanajuato, Mexico in 2019. His research interests include design and analysis of algorithms, swarm robotics, discrete applied mathematics, and data science.

Rafael Murrieta-Cid received the B.S degree in physics engineering from the Monterrey Institute of Technology and Higher Education, Monterrey, Mexico, in 1990 and the Ph.D. degree in robotics (visual navigation) from the Institut National Polytechnique, Toulouse, France, in 1998. His Ph.D. research was done with the Robotics Group (RIA) of the LAAS-CNRS. In 1998-1999, he was a Postdoctoral Researcher with the Department of Computer Science, Stanford University, Stanford CA, USA. During 2002-2004, he was a Postdoctoral Researcher with the Beckman Institute and the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign (UIUC), Urbana, IL, USA. Since 2006, he has been with the Centro de Investigacion en Matematicas CIMAT, in Guanajuato, where he holds the position of senior research scientist (Investigador Titular C), he is member of the Mexican National System of Researchers (SNI), rank 3, and he is an IEEE Transactions on Robotics (T-RO) Associate Editor. In 2016, he was on a sabbatical leave at UIUC. His research interests include robotics, robot motion planning and control theory.