Adaptive target tracking with a mixed team of static and mobile guards: deployment and activation strategies

Guillermo J. Laguna & Sourabh Bhattacharya

Autonomous Robots

ISSN 0929-5593

Auton Robot
DOI 10.1007/s10514-019-09833-8

Autonomous Robots
https://doi.org/10.1007/s10514-019-09833-8

Adaptive target tracking with a mixed team of static and mobile guards: deployment and activation strategies

Guillermo J. Laguna[1] · Sourabh Bhattacharya[1,2]

Received: 1 May 2018 / Accepted: 17 January 2019

Abstract
This work explores a variation of the art gallery problem in which a team of static and mobile guards track a mobile intruder with unknown maximum speed. We consider the special case when the mobile guards are restricted to move along the diagonals of a polygonal environment. First, we present an algorithm to identify *candidate vertices* in a polygon at which either static guards can be placed or they can serve as an endpoint of the segment on which mobile guards move. Next, we present a technique to partition the environment based on the triangulation of the environment, and allocate guards to each partition to track the intruder. The allocation strategy leads to a classification of the mobile guards based on their task and coordination requirements. Finally, we present a strategy to activate/deactivate static guards based on the speed of the intruder. Simulation results are presented to validate the efficacy of the proposed techniques.

Keywords Target tracking · Mobile coverage · Sensor networks

1 Introduction

In the last two decades, there has been a widespread deployment of multi-robot systems as sensor platforms for gaining situational awareness in military (Simon et al. 2004) as well as civilian (Werner-Allen et al. 2006) applications. Robots deployed in such scenarios are capable of autonomously making decisions and taking actions with minimal human intervention. Prior information regarding the exogenous processes that effect the performance of such robotic systems is an essential component of their decision-making process (Theodoridis and Hu 2012). However, in reality, the system normally does not have a complete knowledge about the uncertainty in the environment, especially when adversaries are involved. Due to the sensitive nature of applications, a system designer either uses a Bayesian approach (Ho and Lee 1964) to maximize the expected reward or a game-theoretic approach (Başar and Bernhard 2008) to guarantee a minimum reward in such scenarios. In this work, we explore a different direction, i.e., an *adaptive* approach to address the challenge that arises due to incomplete information. To be specific, we consider the problem of tracking a mobile intruder in an environment with unknown maximum speed using a team of static amd mobile sensors.

Tracking an extremely fast intruder inside a simple polygonal environment requires covering the polygon for which $\lfloor n/3 \rfloor$ static guards are sufficient (and sometimes necessary) (O'Rourke 1987), where n denotes the number of edges of the polygon. At low intruder speeds, Laguna et al. (2016) proposes a tracking strategy [based on mobile coverage (O'Rourke 1983)] for which $\lfloor n/4 \rfloor$ diagonal guards are sufficient. Our current work generalizes the aforementioned results, and presents a strategy to track an intruder with arbitrary speed without covering the entire environment at all times. We leverage results on coverage using static and mobile guards (O'Rourke 1983) to develop deployment and activation strategy for a mixed team of static and mobile sensors to track the mobile intruder. The deployment strategy guarantees robustness to tracking performance in the face of uncertainty (lack of knowledge to be precise) about

This is one of the several papers published in *Autonomous Robots* comprising the Special Issue on Multi-Robot and Multi-Agent Systems.

✉ Sourabh Bhattacharya
sbhattac@iastate.edu

Guillermo J. Laguna
gjlaguna@iastate.edu

1 Department of Mechanical Engineering, Iowa State University, Ames, IA, USA

2 Department of Computer Science, Iowa State University, Ames, IA, USA

Published online: 04 February 2019

the intruder's maximum speed. Nevertheless, the activation strategy for the sensors makes it adaptive to the intruder's instantaneous speed, thereby ensuring tracking without trivially covering the environment at all times.

The contributions of this work are as follows: (1) To the best of our knowledge, this is the first work that addresses the problem of persistent tracking of an intruder without *a priori* knowledge about its maximum speed. (2) We propose a tracking strategy that is *adaptive* i.e., it takes into consideration the instantaneous speed of the intruder to activate the sensors. Although, there are several works in sensor networks and control that address the problem of controlled sensing, this is the first work that addresses controlled sensing in the context of target tracking. (3) The problem of finding the minimum number of sensors required to track an intruder is NP-hard (Laguna and Bhattacharya 2017). To the best of our knowledge, this is the first work that provides results on the sufficient number of sensors required to track an intruder in an environment without trivially covering it, thereby, reducing the bound from $\lfloor n/3 \rfloor$.

The paper is organized as follows. Section 2 presents a review of the related work. In Sect. 3, we present the problem formulation. In Sect. 4, we present the deployment strategy for the guards. In Sect. 5, we present a classification of the guards, and their tracking strategy. In Sect. 6, we present the activation strategy for the static guards. In Sect. 7, we present the results of the proposed technique for two different polygonal environments. Section 8 presents the conclusions and future work.

2 Related work

Target tracking refers to the problem of planning collision-free paths for a mobile sensor that tries to keep a mobile target in its field-of-view. A detailed review regarding prior work related to tracking is provided in Bhattacharya and Hutchinson (2010) and Bhattacharya and Hutchinson (2011). For multiple mobile observers, centralized (Mehmetcik and Ozguner 2013) as well as decentralized techniques (Luke et al. 2005) have been proposed to track a single target (Hausman et al. 2015; Williams and Sukhatme 2015) or a team of targets (Murrieta-Cid et al. 2002; Zou and Bhattacharya 2015; Jung and Sukhatme 2002). In contradistinction to the previous works, this paper assumes that an observer is either static or restricted to move along a pre-specified path.

Without knowledge about the target's current or future actions, the tracking problem gives rise to a visibility-based pursuit-evasion game. In the past, there have been several efforts to obtain optimal strategies for the observer by formulating the tracking problem as a game between the observer, modeled as a pursuer, and the target, modeled as an evader (Bhattacharya and Hutchinson 2010; Bhattacharya et al. 2014). A related problem that has adopted a similar approach is that of *target search*. The objective of target search is to deploy a team of mobile robots to locate a target inside an environment (LaValle et al. 1997; Suzuki and Yamashita 1992). In case of multiple searchers, one tries to find the minimum number of searchers required to *capture* a moving target in a partitioned environment represented as a graph (Megiddo et al. 1981; Isler et al. 2004). In contrast, our current work assumes that the target is initially visible to at least one observer, and the goal for the team of observers is to ensure that it will be visible to at least one observer at all times in the future.

In persistent long-term surveillance, an important challenge for a network designer is to balance the trade-off between energy consumption of the network (Bhatti and Xu 2009) and its overall coverage (Howard et al. 2002). Energy efficiency is an integral component in the design of sensor networks to increase their average lifetime. A frequently used approach to extend the average lifetime of a sensor network is to schedule activation cycles of the individual sensors (He and Chong 2006). For example, Tian and Georganas (2002) proposes scheduling algorithms that randomly assign sensor nodes to multiple working subsets, which alternately perform the sensing tasks for monitoring a sensor field. In Deng et al. (2005), authors propose a scheduling algorithm for clustered sensor networks that deactivates nodes based on their relative distance to the cluster head to improve energy saving. In addition to energy efficiency, scheduling schemes often need to fulfill a secondary objective, for example, preserving the transmission range (Chen et al. 2007) or the sensing coverage (Kumar et al. 2004) of the network. For a static camera network, reducing the volume of data collected is an important secondary objective since it lowers the overall latency of the network (Aghajan and Cavallaro 2009) by reducing the communication and processing overheads. A mobile camera network can alleviate this problem of *data deluge* (Baraniuk 2011) by reducing the number of sensors that need to be active. In this work, we consider a camera network for a persistent target tracking problem. To achieve our task while reducing the latency and energy consumption of the network, a collaborative scheme between a team of mobile sensors and a team of static sensors is proposed.

In this work, we leverage results from *art gallery problems* (Chvatal 1975). The art gallery problem originates from a real-world problem of guarding an art gallery with a team of guards who together can observe the whole gallery. For an infinite speed intruder, the problem of persistently tracking it reduces to a coverage problem. Results from the art-gallery problem can be used to deploy static guards (O'Rourke 1987; Hoffmann 1990) in a polygonal environment to ensure coverage. However, for the case of an intruder with finite speed, fewer guards may be sufficient to track it. In Laguna et al. (2016) and Laguna and Bhattacharya (2017), we proposed a tracking strategy that deploys at most $\lfloor \frac{n}{4} \rfloor$ diagonal guards in a simple polygon with n sides, and derived the maximum speed of the intruder for which the tracking strategy works for a fixed speed of the guards. All the aforementioned works assume that the maximum speed of the intruder is known *a priori* which is a severe limitation since knowledge about the adversary is limited in surveillance applications. In contrast, this paper deals with the scenario when the guards do not have any *a priori* knowledge about the maximum speed of the intruder.

3 Problem formulation and motivation

For a polygon P, a *triangulation* of P is defined as a (not necessarily unique) partition of P into a set of disjoint triangles such that the vertices of the triangles are vertices of P, and there is no intersection between any pair of edges of the triangles. An edge of a triangle in the triangulation is called a *diagonal* (O'Rourke 1983). The triangulation of a polygon P can be represented as a planar graph $G = G(P)$ called a *triangulation graph*. $V(G)$ (vertex set of G) corresponds to the vertices of P, $E(G)$ (edge set of G) corresponds to the diagonals of the triangulation of P, and $T(G)$ (triangle set of G) corresponds to the faces[1] of G. Clearly, there is a bijection between the set of vertices of P and $V(G)$, and between the set of diagonals of the triangulation of P and $E(G)$. As a result, we do not make a distinction between the following: (1) vertices of P and the vertices in $V(G)$ (2) diagonals of the triangulation of P and the edges in $E(G)$ (3) triangles of the triangulation of P and the triangles in $T(G)$. Let D be the *dual graph* of G. Thus, each vertex in $V(D)$ corresponds to a triangle in $T(G)$, and for each $e_{i,j} \in E(D)$, where $v_i, v_j \in V(D)$, there is a pair of triangles $T_i, T_j \in T(G)$ that share an edge. For any triangulation graph of a simple polygon, the dual graph is a tree (Berg et al. 2008).

3.1 Problem formulation

Consider a closed environment that can be represented as a simple n-gon (n edges/vertices) P with no holes. A mobile intruder I lies inside the polygon. At a time t, I is located at $p_I(t) \in P$, and it moves at a speed $v_I(t) \in [0, \mathbf{v}_I^{max}]$, where $\mathbf{v}_I^{max}$ denotes its maximum speed. We define $S_g = S_g^h \cup S_g^v$ as a set of guards deployed in the environment, where S_g^h and S_g^v denote the sets corresponding to mobile and vertex guards respectively, and $S_g^h \cap S_g^v = \emptyset$. We assume that each $g_i \in S_g$ is equipped with an omni directional vision sensor with infinite range. As a result, g_i can see everything that lies inside its visibility polygon (Berg et al. 2008). A region inside the polygon is said to be *covered* by the guard if it lies inside the visibility polygon of the guard. Next, we provide a description of the static and mobile guards.

The static guards are located at the vertices of the polygon. Each $g_i \in S_g^v$ is located at a vertex $v \in V(G)$ and is always static. For convenience and to remark the correspondence between the vertex and the guard, we call v_i to the vertex where g_i is located. Moreover, each $g_i \in S_g^v$ can either be in an *active mode* (consuming power with its camera switched on) or in *sleep mode* (negligible power consumption with its camera switched off). g_i is said to be activated (deactivated) when it goes from the sleep mode to the active mode (active mode to the sleep mode).

In this work, we consider mobile guards to be *diagonal guards* (O'Rourke 1983) i.e., each $g_i \in S_g^h$ is constrained to move along a diagonal $h_i \in E(G)$. The advantage of considering diagonal guards is twofold: (i) It can be easily implemented on a linearly actuated sliding camera system (Durocher and Mehrabi 2013) (2) It allows us to leverage results in mobile coverage (O'Rourke 1983). Two diagonal guards do not share their paths, so for each $g_i, g_j \in S_g^h$ with $i \neq j$, $h_i \neq h_j$. The endpoints of h_i are denoted by $v_j(i)$ with $j \in \{1, 2\}$, and its length by l_i. Each $g_i \in S_g^h$ is assumed to have a maximum speed $0 < \mathbf{v}_g^{max} < \infty$. Let $r = \mathbf{v}_I / \mathbf{v}_g$ denote the *speed ratio*. In contrast to the static/vertex guards, the mobile guards are assumed to be always in an active mode. Therefore, they are always ready to move.

A trivial solution to the problem of tracking is to cover the entire environment with static guards at all times. However, it might be possible to track the intruder with fewer guards depending on its instantaneous speed. Therefore, the problem is to develop a tracking strategy for the team of guards that is *adaptive* to the instantaneous speed of the intruder. One way to achieve this adaptation is to activate/deactivate guards based on the instantaneous speed of the intruder. Since the maximum speed of the intruder is not known to the guards, they should be able to cover the environment in the worst case scenario (as $v_I(t) \longrightarrow \infty$).

Our proposed solution is to deploy a team of static and diagonal guards to track the intruder. The motion control of the diagonal guards, and the activation strategy of the static guards depend on the instantaneous speed of the intruder. Leveraging on ideas from mobile coverage in art gallery problems, the diagonal guards can track the intruder at low speeds (Laguna and Bhattacharya 2017) while the static guards remain inactive. As the speed of the intruder increases beyond a threshold (that can be computed from Laguna and Bhattacharya (2017)), static guards are activated. For the limiting case in which the speed of the intruder approaches infinity, the diagonal guards stay at one end of the diagonal (i.e, become vertex guards), and all the guards cover the entire environment.

[1] In a triangulation graph all the faces are triangles.

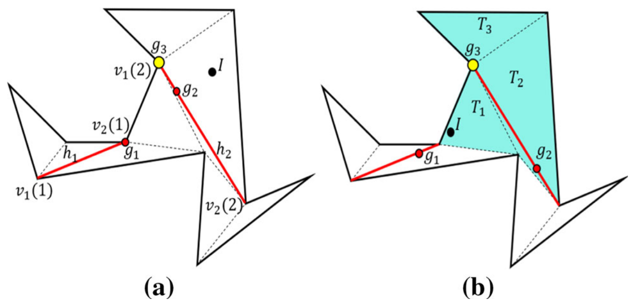


Fig. 1 **a** g_1 and g_2 are tracking I while g_3 is a deactivated vertex guard. **b** Activation of g_3 to aid g_1 and g_2

Figure 1 illustrates an example to motivate our problem. Figure 1a shows a triangulated polygon. Circles shaded in red represent guards (g_1 and g_2), and the red segments represent the diagonals allocated to the guards (h_1 and h_2, respectively). The circle shaded in black denotes the intruder I. The circle shaded in yellow is a vertex guard g_3 in sleep mode, which is located at endpoint $v_1(2)$ of h_2. g_1 and g_2 are sufficient to track the intruder at small values of $\mathbf{v}_I$. However, beyond a critical value of $\mathbf{v}_I$, g_3 needs to be activated to track the intruder since the maximum speed of g_1 and g_2 may not be sufficient to guarantee persistent tracking. This is illustrated in Fig. 1b where we assume that a sufficiently fast intruder can "hide" inside triangle T_1. In that case, the inactive vertex guard g_3 is activated, from its location it covers triangles T_1, T_2 and T_3.

In the next section, we present a deployment strategy for the guards.

4 Deployment of guards

From O'Rourke (1987), we know that for every polygon there exists a set of dominating vertices[2] and dominating diagonals[3] of size at most $\lfloor n/3 \rfloor$ and $\lfloor n/4 \rfloor$, respectively. Placing a static guard at the dominating vertices can provide static coverage. On the other hand, placing a mobile guard on the dominating diagonals can provide mobile coverage. The sets of dominating vertices and dominating diagonals for a polygon is not unique. In this section, we show that for every polygon there exists a set S_c of at most $\lfloor n/3 \rfloor$ dominating vertices which contains a subset of (at most $\lfloor n/4 \rfloor$) vertices, such that each one of the vertices of the subset is the endpoint of a diagonal in a set of at most $\lfloor n/4 \rfloor$ dominating diagonals. We call such a set S_c, a set of *candidate vertices*. A point to note here is that the set of at most $\lfloor n/4 \rfloor$ candidate vertices S_c depends on the triangulation of P. As it was mentioned in Sect. 3, the triangulation of a polygon might not be unique. There are several methods to obtain a triangulation of a polygon (O'Rourke 1987). Here we assume that the triangulation is already given. The problem of selecting a "good" triangulation that leads to a smaller number of candidate vertices is a part of our future research.

We present the following results from O'Rourke (1987) which are used to prove Lemma 1:

1. In any triangulation graph G of a simple polygon P with $n \geq 10$ edges it is always possible to find a diagonal $d \in E(G)$ that partitions P into P_1 and P_2 such that P_1 is the triangulation graph of a pentagon, hexagon, heptagon, octagon or nonagon, which we call *basic polygons*.
2. Any triangulation graph of a pentagon, hexagon or heptagon has one dominating diagonal.
3. Any triangulation graph of a nonagon has two dominating diagonals, and in general every triangulation graph of a polygon with $n \geq 5$ vertices has a set of dominating diagonals with size of at most $\lfloor n/4 \rfloor$.

Remark 1 Lemma 1 and Theorem 1 prove the existence of a set of at most $\lfloor \frac{n}{3} \rfloor$ dominating vertices of $V(G)$, and a set of dominating diagonals of size of at most $\lfloor \frac{n}{4} \rfloor$ such that each diagonal in the set of dominating diagonals has a different vertex in the set of dominating vertices as an endpoint. In contrast, Theorem 1 in O'Rourke (1983) proves the existence of dominating diagonals in a polygon. Although, proofs of Lemma 1 and Theorem 1 are inductive, the intermediate steps are different from the proof of Theorem 1 in O'Rourke (1983).

Lemma 1 *For any given triangulation graph of a basic polygon, there exists a set S_c of at most $\lfloor n/3 \rfloor$ dominating vertices and a set S_h of at most $\lfloor n/4 \rfloor$ dominating diagonals, such that an endpoint of each $h_i \in S_h$ is a $v_i \in S_c$, and for each $h_i, h_j \in S_h$, $v_i \neq v_j$ for $i \neq j$. (There exists a set of candidate vertices).*

Proof 1. For a pentagon, the proof of the lemma follows from the proof of O'Rourke (1987) (Lemma 3.3, page 85) which states that there is always a vertex that can dominate any triangulation graph of a pentagon, so it can always be dominated by one diagonal guard with such a vertex as an endpoint of its diagonal while the opposite endpoint can be any other vertex of the triangulation. Hence, $|S_c| = |S_h| = 1 \leq \lfloor 5/3 \rfloor = \lfloor 5/4 \rfloor$.

2. Irrespective of the triangulation of the hexagon, it can always be decomposed into a pentagon P_1 and a triangle T_1 such that T_1 can be dominated by a vertex v_1 which is an arbitrary vertex of T_1. Therefore, a candidate vertex v_0 exists from which P_1 can be dominated according to

[2] A triangulation graph G is said to be *dominated by a set of vertices* S_c if at least one vertex of each triangle in $T(G)$ is a vertex in S_c.

[3] A triangulation graph G is said to be *dominated by a set of diagonals* S_h if at least one vertex of each triangle in $T(G)$ is an endpoint of a diagonal in S_h.

the case of a pentagon. We can always select a diagonal with an endpoint at v_0 and the other at one vertex of T_1. Thus, $|S_c| \leq 2 = \lfloor 6/3 \rfloor$ and $|S_h| = 1 = \lfloor 6/4 \rfloor$.

3. Regardless of its triangulation, a heptagon can always be decomposed into a hexagon H_1 and a triangle T_1. Based on the case of a hexagon, there are two candidate vertices v_0 and v_1 from which H_1 can be dominated. Since v_1 can be an arbitrary vertex, a vertex of T_1 can always be chosen to cover it. v_1 is a candidate vertex on an arbitrary edge. The diagonal is selected such that one of its endpoints is v_0, and its other endpoint is a vertex of T_1. Thus, $|S_c| \leq 2 = \lfloor 7/3 \rfloor$ and $|S_h| = 1 = \lfloor 7/4 \rfloor$.
4. Regardless of its triangulation, an octagon can always be decomposed into a heptagon H_2 and a triangle T_1. Based on the case of a heptagon, there are two candidate vertices v_0 and v_1 from which H_2 can be dominated. Since v_1 can be a vertex of an arbitrary edge of H_2, one of the endpoints of the edge shared by H_2 and T_1 can always be chosen as a candidate vertex. One diagonal is selected such that one of its endpoints is v_0 and it dominates H_2. If it cannot dominate T_1, any diagonal with v_1 as an endpoint can be chosen to dominate T_1. Thus, $|S_c| \leq 2 = \lfloor 8/3 \rfloor$ and $|S_h| \leq 2 = \lfloor 8/4 \rfloor$.
5. Regardless of its triangulation, a nonagon can always be decomposed into the triangulation graph of an octagon O_1 and a triangle T_1. From the case of an octagon, there are two candidate vertices v_0 and v_1 that dominate O_1. Hence, an additional guard v_2 is required to cover T_1. Since v_1 can be an arbitrary vertex of T_1 which in turn can be arbitrarily chosen to include any vertex of the boundary of the nonagon, it follows that v_1 can be arbitrarily placed, and the remaining triangulation subgraph of the nonagon has two candidate vertices. A diagonal incident[4] to v_2 can dominate at least two triangles of G. The remaining subgraph of G is the triangulation graph of a heptagon with two candidate vertices (v_0 and v_1), and it can be dominated by a single diagonal guard with an endpoint incident to v_0 or v_1 (the case of a heptagon). Thus, $|S_c| \leq 3 = \lfloor 9/3 \rfloor$ and $|S_h| \leq 2 = \lfloor 9/4 \rfloor$. □

Theorem 1 *There is a set of candidate vertices for any triangulation graph G of a polygon with $n \geq 5$ vertices.*

Proof Lemma 1 proves the statement of the theorem for $5 \leq n \leq 9$. Therefore, we prove the theorem for the case when $n \geq 10$ using induction. Let us assume that the theorem holds for any $n' < n$ and $n \geq 10$. In O'Rourke (1987), it is shown that a diagonal d that partitions G into two graphs G_1 and G_2, where G_1 contains k boundary edges with $4 \leq k \leq 8$, always exist. We consider each value of $4 < k \leq 8$:

1. $k = 5$: G_1 has $k+1 = 6$ boundary edges including d. By Lemma 1, G_1 has two candidate vertices v_0 and v_1, with v_1 located arbitrarily. There is a dominating diagonal with v_0 or v_1 as an endpoint. G_2 has $n-k+1 = n-4$ boundary edges including d, and since v_1 is arbitrary, it can be an endpoint of the edge between G_1 and G_2. Hence, we can always select v_1 such that at least two triangles T_a and T_b of G_2 are incident to v_1. Let $S_v(T_a \cup T_b)$ and $S_e(T_a \cup T_b)$ be the sets of vertices and edges of the union of T_a and T_b respectively. A graph G_0 can be constructed such that $V(G_0) = V(G_1) \cup S_v(T_a \cup T_b)$ and $E(G_0) = E(G_1) \cup S_e(T_a \cup T_b)$ (we say that T_a and T_b were adjoined to G_1), G_2 has now $n-6$ vertices which implies that it has at most $\lfloor n/3 \rfloor - 2$ candidate vertices. Thus, G_0 together with G_2 has at most $\lfloor n/3 \rfloor$ candidate vertices.
2. $k = 6$: This is similar to the previous case, but G_1 has $k+1 = 7$ boundary edges and v_1 is the endpoint of an arbitrary edge. G_2 has $n-k+1 = n-5$ boundary edges including d. Since v_1 is the endpoint of an arbitrary edge, it can belong to the edge between G_1 and G_2. Hence, v_1 is selected such that it dominates at least one triangle of G_2. G_0 is obtained by adjoining this triangle to G_1, so G_2 has $n-6$ vertices. Therefore, it has at most $\lfloor n/3 \rfloor - 2$ candidate vertices. The result follows.
3. $k = 7$: G_1 has $k+1 = 8$ boundary edges. G_2 has $n-7+1 = n-6$ boundary edges. In O'Rourke (1983) (Theorem 1,page 280) there are two cases: (1) the triangulation graph of the octagon has a single dominating diagonal. Therefore, v_0 and v_1 can be the endpoints of the diagonal. G_2 has $n-6$ boundary edges. Hence, it has $\lfloor n/3 - 2 \rfloor$ candidate vertices, and G has at most $\lfloor n/3 \rfloor$ candidate vertices. (2) A graph G_3 is obtained by adjoining three triangles of the triangulation graph of the octagon to G_2. The remaining subgraph corresponds to a pentagon. Since G_3 has $n-4+1 = n-3$ vertices, it has $\lfloor n/3 \rfloor - 1$ candidate vertices. Therefore, G has at most $\lfloor n/3 \rfloor$ candidate vertices since the triangulation of the remaining pentagon has only one candidate vertex 1.
4. $k = 8$: G_1 has $k+1 = 9$ boundary edges, and by Lemma 1, it has three candidate vertices v_0, v_1 and v_2 that dominate G such that v_2 is an arbitrary endpoint of the edge between G_1 and G_2, and it is an endpoint of one of the two dominating diagonals. v_2 can be chosen to cover at least two triangles of G_2. G_0 is obtained by adjoining those two triangles to G_1. Consequently, G_2 has $n-9$ boundary edges. Hence, G_2 has $\lfloor n/3 \rfloor - 3$ candidate vertices, it follows that G has at most $\lfloor n/3 \rfloor$ candidate vertices.

It is guaranteed that for any simple polygon with $n \geq 10$ vertices, there is always a diagonal d that partitions the triangulation graph of the polygon such that the partition corresponds to one of the aforementioned cases. Since for

[4] A diagonal is said to be *incident* to a vertex if the vertex is an endpoint of the diagonal.

each one of such cases it was shown that there exists a set of candidate vertices, and by Lemma 1 we know that the statement is always true for $4 < n < 10$, it follows that there is a set of candidate vertices for any triangulation graph of a simple polygon with $n > 4$ vertices. □

Algorithm 1 Guard Deployment

```
1: Input: $G$.
2: Output: $S_h$ and $S_c$
3: $S_d \leftarrow \emptyset$ is the set of diagonals $d$
4: $G' \leftarrow G$
5: while $G'$ has $n \geq 10$ vertices do
6:     find $d$ that separates the triangulation of a minimal basic polygon $G_k$ from $G'$
7:     $G'$ becomes the subgraph obtained by removing $G_k$ excepting the vertices of $d$
8:     add $d$ to $S_D$
9: end while
10: create $G_{pol}$ from $G$ using the diagonals in $S_d$
11: while there is an unmarked vertex in $G_{pol}$ do
12:     $v_k \leftarrow$ unmarked vertex in $G_{pol}$ with at most one unmarked neighbor
13:     $G_k \leftarrow$ subgraph of $G$ that corresponds to $v_k$
14:     $s_c \leftarrow$ appropriate candidate vertices of $G_k$
15:     $s_h \leftarrow$ appropriate diagonals of $G_k$
16:     add vertices of $s_c$ to $S_c$
17:     add diagonals of $s_h$ to $S_h$
18:     mark $v_k$
19: end while
```

Theorem 1 proves the existence of a set of candidate vertices S_c in $V(G)$ (and the existence of the corresponding set of dominating diagonals S_h). Based on the different cases of the theorem and the existence of the diagonals that partition G into subgraphs corresponding to basic polygons Algorithm 1 is an exhaustive strategy to determine S_c and S_h. Algorithm 1 iteratively partitions G into triangulation subgraphs of basic polygons, and for each one of those subgraphs the sets of candidate vertices and dominating diagonals are found exhaustively. At each iteration, Algorithm 1 searches a diagonal d that separates a triangulation subgraph of a basic polygon (denoted by G_p) such that there is no other diagonal in $E(G_p)$ that can separate a subgraph of a smaller basic polygon. The process is repeated until the remaining non-partitioned subgraph has 9 vertices or less. The first **while** cycle (Line 5) finds the diagonal d that partition G into triangulation subgraphs that correspond to basic polygons. This can be completed in $O(n)$ time by traversing the dual graph G_D. Recall that $|V(D)| = n - 2$ and D is a tree. In Fig. 2a a simple polygon is shown, it has $n = 18$ vertices and 16 triangles, the green segments represent the diagonals d_1, d_2 and d_3 found by Algorithm 1. d_1 separates the triangulation graph of the hexagon that contains triangles T_5, T_6, T_7 and T_8. d_2 (along with d_1) separates an hexagon containing T_1, T_2, T_3 and T_4. Finally, d_3 is found, which separates a heptagon

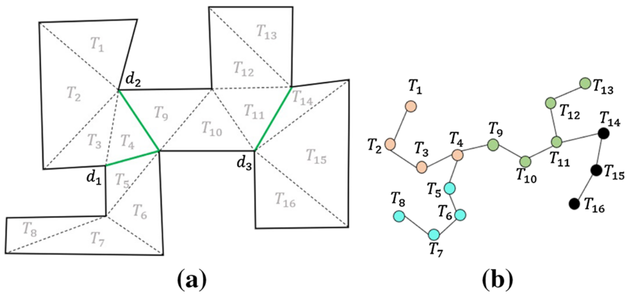


Fig. 2 **a** Simple polygon partitioned into triangulations of basic polygons. **b** Corresponding dual graph

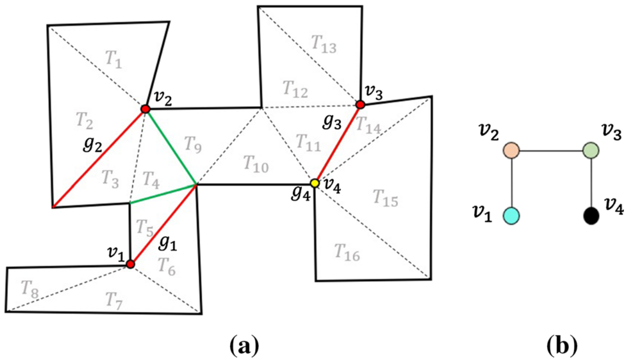


Fig. 3 **a** Candidate vertices and dominating diagonals. **b** Corresponding graph G_{pol}

containing triangles T_9, T_{10}, T_{11}, T_{12} and T_{13}. The remaining subgraph corresponds to a pentagon. Figure 2b shows the corresponding dual graph, the blue vertices correspond to the triangles of the hexagon separated by d_1, the orange vertices to the triangles separated by d_2, and the green vertices to the triangles separated by d_3.

After the first while loop, a graph G_{pol} is created (line 10) such that each vertex $v \in V(G_{pol})$ corresponds to the triangulation graph of each basic polygon G_k that was found in line 6, where k stands for the k_{th} basic polygon found, and there is an edge $e \in E(G_{pol})$ only between vertices that correspond to basic polygons that have a common edge. For this step, D along with S_d can be used to identify in linear time the triangles in $T(G)$ that correspond to the basic polygons of the partition, and consequently their subgraphs G_k. This takes $O(n)$ time. In Fig. 3b the graph G_{pol} that corresponds to the example of Fig. 2 is shown, the blue and orange vertices correspond to the hexagons found in the partition, the green vertex corresponds to the heptagon and the black vertex corresponds to the remaining pentagon. The second while loop identifies the minimum set of vertices that can cover the triangles in $T(G_k)$ that are not already covered by other candidate vertices (line 14). Finding G_k takes $O(n)$ time.

We are interested in the set of candidate vertices of G_k and its corresponding set of dominating diagonals. Since the sets of candidate vertices and diagonals of neighboring subgraphs might have already been identified in the previous steps by the algorithm, it is possible that some triangles in G_k are already dominated either by candidate vertices or by dominating diagonals. Therefore, to find the sets of candidate vertices and dominating diagonals for G_k, instead of considering all the triangles in G_k, we only need to consider the triangles in G_k that are not dominated by candidate vertices and the triangles that are not dominated by dominating diagonals, respectively. Notice that an undominated triangle can be "undominated by diagonals", which means that there is no dominating diagonal that dominates it, or can be "undominated by vertices", which means that there is no candidate vertex dominating it. Thus, when we talk about candidate vertices that can dominate undominated triangles we refer only to triangles undominated by vertices. An equivalent statement is made when referring to dominating diagonals and undominated triangles. The set of candidate vertices must ensure that no undominated triangles (by vertices) remain, while the set of dominating diagonals must ensure that no undominated triangles (by diagonals) remain. The set of candidate vertices of G_k is upper bounded by $\lfloor |V(G_k)|/3 \rfloor$ by definition, and a subset of those vertices are endpoints of a set at most $\lfloor |V(G_k)|/4 \rfloor$ dominating diagonals, where $|V(G_k)|$ is the number of vertices of G_k. From Theorem 1, we know that such a pair of sets exist which implies that there is a set of vertices in $V(G_k)$ that meets the definition of a set of candidate vertices. We consider the pairs of dominating diagonals and candidate vertices for which the number of dominating diagonals is minimum. In case of a tie among the the pairs of such sets, we select the subsets for which the corresponding number of candidate vertices is minimum. In case of a tie, then we select the subsets of pairs for which the set of dominating diagonals dominates the maximum number of undominated triangles (not in $T(G_k)$). In the case of a tie, the pair of sets where the set of candidate vertices dominates the maximum number of undominated triangles (not in $T(G_k)$). Finally, if a tie still exists, the selection of the pair of sets is arbitrary among the pairs of sets that qualify. The number of combinations to be considered is determined by the number of vertices in $V(G_k)$. Thus, in the worst case, which occurs when G_k is a nonagon $\implies |V(G_k)| = 9$ and $|E(G_k)| = 15$. The algorithm then needs to try all the 135 combinations of pairs of edges of G_k ($\lfloor 9/4 \rfloor = 2$) and no more than 84 combinations of groups of 3 vertices. For any G_k with fewer vertices the number of combinations to check is smaller. The number of combinations to check is fixed in each case. Therefore, finding the candidate vertices for G_k takes constant time.

Since $|V(G_{pol})| < n$, the second **while** loop takes $O(n)$ time. It follows that the time complexity of the algorithm is $O(n^2)$. The procedure to mark the visited vertices in $V(G_{pol})$ ensures that at any time a triangulation subgraph is selected by the algorithm to find its corresponding candidate vertices and dominating diagonals, there is at most one neighboring vertex in $V(G_{pol})$ that corresponds to a triangulation subgraph with non-dominated triangles. Once that S_h and S_c are found, guards in S_g^h are deployed in the diagonals of S_h while guards in S_g^v are deployed in the vertices of S_c that are not associated to the diagonals in S_h.

In Fig. 3a, the red diagonals represent the dominating diagonals, the red shaded circles correspond to the candidate vertices assigned to the dominating diagonals, and the yellow shaded circle corresponds to the remaining candidate vertex which is not associated to any diagonal. Starting with the hexagon defined by triangles T_8, T_7, T_6 and T_5 there is a single vertex from which all the triangles of the hexagon can be covered (v_1). Any diagonal with v_1 as an endpoint dominates the hexagon. However, the diagonal chosen is the one that dominates more triangles (in the example it is the diagonal that dominates T_4 T_9 and T_{10}). Next, the algorithm proceeds with the second hexagon, where v_2 is found and since any diagonal of the hexagon with v_2 as an endpoint dominates the hexagon, the diagonal labeled as g_2 was arbitrarily selected. The algorithm proceeds with the triangulation graph of the heptagon where two candidate vertices v_3 and v_4 are found, and also one diagonal incident to both which is able to dominate the triangles that are not already dominated. Hence, $S_c = \{v_1, v_2, v_3, v_4\}$ and $S_h = \{h_1, h_2, h_3\}$. Clearly, $|S_c| < \lfloor 18/3 \rfloor$ and $|S_h| < \lfloor 18/4 \rfloor$. Finally, three diagonal guards g_1, g_2 and g_3 are deployed along h_1, h_2 and h_3 respectively, and a vertex guard g_4 is located at v_4.

5 Tracking with diagonal guards

In this section we present a strategy for the mobile guards to track the intruder. Let $g_i \in S_g^h$, and let $T_j(i) \subset T(G)$ be the set of triangles incident to $v_j(i)$ with $j \in \{1, 2\}$. We say that the set of triangles $T(i) = T_1(i) \cup T_2(i)$ is "incident" to h_i. We define $d_M^i = rl_i$ as the "Maximum" distance that the intruder can travel while g_i moves across h_i at maximum speed. The triangles of $T(G)$ can be classified in the following categories based on the relative location of the diagonals of S_h (refer to Fig. 4):

1. A triangle is called *safe* if it can be covered at any time the intruder is inside it. They are illustrated as light blue shaded triangles in Fig. 4. Trivially, a triangle such that one or more of its edges is a diagonal in S_h, or a triangle such that there is an active static guard located at any of its vertices is a safe triangle. In Fig. 4, g_3 is a static guard located at the vertex represented as a red circle and all the triangles that have such a vertex in common are safe triangles.

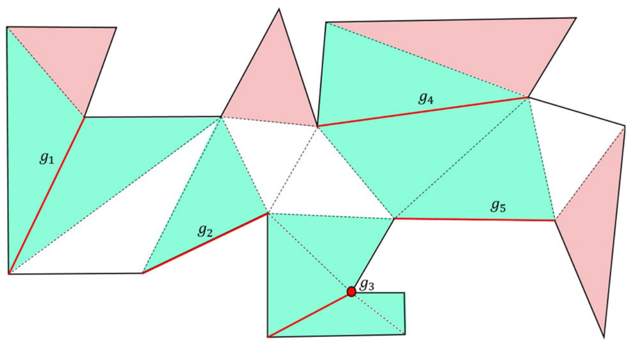


Fig. 4 Classification of triangles

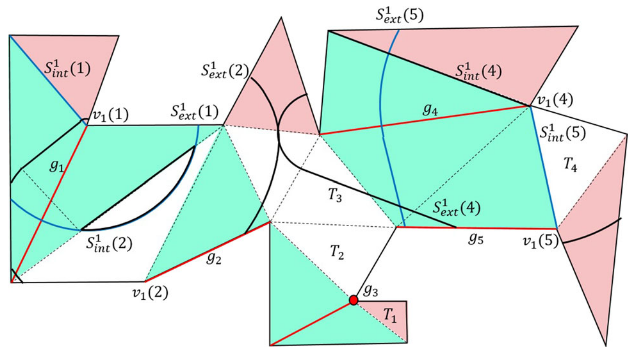


Fig. 5 g_1 is a Type 1 guard, g_2 is a Type 2 guard, g_3 is a Type 0 guard, and g_4 and g_5 are Type 5 guards

2. A triangle is called *unsafe* if it is not a safe triangle and there is only one diagonal guard that can cover it from one of its endpoints. They are illustrated as orange shaded triangles in Fig. 4.
3. A triangle is called *regular* if it is neither safe nor unsafe. They are illustrated as unshaded triangles in Fig. 4. Notice that for each regular triangle there are at least two dominating diagonals (red segments) incident to its vertices.
4. Safe Zone: A *safe zone* of a guard g_i, denoted by $A(i)$, is the set of adjacent triangles which have h_i as an edge.
5. Augmented Safe Zone: An augmented safe zone of a guard g_i, denoted by $B(i)$ is the largest set of adjacent safe triangles in $T(i)$ such that $A(i) \subseteq B(i)$. See Fig. 4, where for each one of the diagonal guards there is an unsafe zone. Notice that for guards g_1 and g_2, $B(i) = A(i)$.
6. Unsafe Zone: An *unsafe zone* of an endpoint $v_j(i)$ of h_i,is the set of unsafe triangles incident to $v_j(i)$ and it is denoted by $U_j(i)$. Also let $U(i) = U_1(i) \cup U_2(i)$ be the unsafe zone of g_i.

Based on the above definitions, we can classify guards into the following categories:

1. Type 0 Guards: A guard $g_i \in S_g$ is of type 0 if there is an endpoint of h_i such that all the triangles incident to such endpoint are safe triangles. Hence, a g_i does not require to be located at one endpoint of its diagonal, so it can just stay motionless at the opposite endpoint, labeled as $v_1(i)$. An example of a Type 0 guard is g_3 in Fig. 5.
2. Type 1 Guards: A guard g_i (not of Type 0) is of Type 1 if it has an unsafe zone at one of the endpoints of h_i, and there are no regular triangles (incident to that endpoint) adjacent to $B(i)$. In such a case, we label such endpoint as $v_1(i)$ (and we label $v_2(i)$ the opposite endpoint). We define an unsafe region as $R = \bigcup_{T_j \in U_1(i)} T_j$. The boundary between $U_1(i)$ and $B(i)$ partitions the interior of P into two sub polygons such that one of the regions of the partition, denoted by P_R contains R. We denote the boundary that partitions P as an internal critical curve $S^1_{int}(i)$. Moreover, we define an external critical curve, $S^1_{ext}(i)$, as the curve inside $P \backslash P_R$ such that the shortest distance inside P between each point in $S^1_{ext}(i)$ and $S^1_{int}(i)$ is equal to d^i_M. Thus, the critical region $C_1(i)$ is defined as the region inside P "enclosed" by $S^1_{int}(i)$ and $S^1_{ext}(i)$. A more detailed definition of this type of guard and its critical region is presented in Laguna et al. (2016). Equation (1) represents the motion strategy of a type 1 guard. It gives the location of g_i based on the location of I when $p_I \in C_1(i)$. Given the definition of $S^1_{ext}(i)$ and d^i_M, the critical curves are used to trigger the motion of g_i such that g_i starts moving from one endpoint of h_i to the other when I enters $C_1(i)$, and it is ensured that g_i can reach the opposite endpoint of h_i at the same time that I reaches the other critical curve bounding $C_1(i)$. A Type 1 guard g_1 is illustrated in Fig. 5 along with its critical curves $S^1_{int}(1)$ and $S^1_{ext}(1)$ shown as blue curves.
3. Type 2 Guards: A guard $g_i \in S_g$ (not of Type 0 or Type 1) is of Type 2 if all the neighboring guards[5] that can cover the regular triangles incident to one endpoint of h_i have their critical regions already defined. Such an endpoint is then labeled as $v_1(i)$. In Fig. 5, g_1 has its critical region defined since it is a Type 1 guard. Moreover, since g_1 is the only neighbor of g_2 that can cover a regular triangle incident to one endpoint of the diagonal of g_2, then g_2 meets the definition of a Type 2 guard. Define $R_1(i) \subset T_1(i)$ as the set of regular triangles incident to $v_1(i)$, and define $N_1(i)$ as the set of all neighboring guards that can cover triangles in $R_1(i)$. We define $S_{T_j} \subset N_1(i)$ as the set of guards that can cover $T_j \in R_1(i)$. Let $B_j = T_j \cap (\bigcap_{g_l \in S_{T_j}} C_1(l))$. Notice that if $B_j \neq \emptyset$ and $p_I \in B_j$, g_i is the only guard that can cover B_j. Let S_B be the region obtained from the union of the regions B_j and all the unsafe triangles incident to $v_1(i)$. We define $S^1_{int}(i)$ as the boundary of S_B. $S^1_{ext}(i)$ and $C_1(i)$ are defined in the

[5] We say that a guard $g_k \in S_g \backslash \{g_i\}$ is a *neighbor* of g_i if $T(i) \cap T(k) \neq \emptyset$.

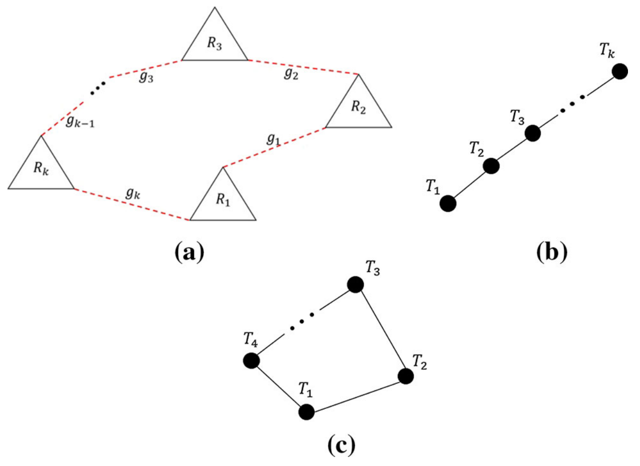


Fig. 6 **a** The presence of type 3 guards means that a closed path between regular triangles and diagonals of guards can be found in G. **b** Graph that represents the recursive process of Algorithm 3. **c** The graph cannot have a closed path

same manner that they are defined for Type 1 guards. In Fig. 5, $S^1_{int}(2)$ is the closed black curve inside the regular triangle shared by g_1 and g_2, and $S^1_{ext}(2)$ is shown as a black curve inside P that maintains a constant distance from $S^1_{int}(2)$.

4. Type 3 Guards: A guard $g_i \in S_g$ is a type 3 guard if it is not of Type 0,1 or 2. A distinctive characteristic of this type is that there is at least one regular triangle incident to each endpoint of h_i and adjacent to $B(i)$, and there is at least a pair of neighbors without their critical regions defined, which means that they also meet the definition of a type 3 guard. Hence, we cannot determine all the regions R that would define its internal critical curve. Since meeting the definition of a Type 3 guard implies the presence of another Type 3 guard that can cover a shared regular triangle, the presence of Type 3 guards implies the existence of a cyclic arrangement between them and regular triangles as shown in Fig. 6a. Clearly, there is an ambiguity in allocation for those regular triangles since the region that must be assigned to g_i cannot be found as in the cases of Type 1 and Type 2 guards. Thus, for Type 3 guards, we proceed by selecting one of them (g_i) arbitrarily and then transforming all the regular triangles incident to one endpoint of its diagonal into unsafe triangles. The selected endpoint is then labeled as $v_1(i)$. This turns g_i into a Type 1 guard, so a critical region for it can be obtained. Consequently, g_i has the task of covering all the triangles in $T_1(i)$. Since all triangles in $R_1(i)$ are unsafe triangles that must be covered by g_i, they are considered as "safe triangles" for the neighboring guards. This change on the types of triangles, transforms the types of the neighboring guards (Type 3 neighboring guards may now meet the definition of one of the other types, Type 2 guards may meet the definition of Type 0 or Type 1 guards, and Type 1 guards may become Type 0 guards). In Fig. 5, two Type 3 guards g_4 and g_5 are illustrated. Triangles T_3 and T_4 are regular triangles shared by both type 3 guards, so it is not possible to obtain the internal critical curve of g_4 and g_5. Hence, T_4 is arbitrarily redefined as an unsafe triangle for g_5 (and consequently as a safe triangle for g_4), that means that it is assigned to g_5. Hence, g_5 meets the definition of a Type 1 guard, and since T_4 is considered a safe triangle by g_4, then it also meets the definition of a Type 1 guard. The resulting critical curves are also illustrated.

5.1 Reactive strategy

Let $p_I \in C_1(i)$ and let $d^1_{min}(i)$ be the minimum distance between p_I and $S^1_{int}(i)$. The following equation maps the location of the intruder to obtain the location of a guard, p_{g_i}, along its diagonal:

$$p_{g_i} = p_{v_1(i)} + \frac{d^1_{min}(i)}{d^i_M}(p_{v_2(i)} - p_{v_1(i)}), \tag{1}$$

where $p_{v_1(i)}$ and $p_{v_2(i)}$ are the coordinates of $v_1(i)$ and $v_2(i)$ respectively. If $p_I \notin C_1(i)$, g_i remains static at $v_1(i)$ or $v_2(i)$ depending on its current location.

We present an important result that determines the condition under which non-safe (regular or unsafe) triangles can be turned into safe triangles. We define $S_g(T)$ as the set of diagonal guards that can cover $T \in T(G)$.

Lemma 2 *A non-safe triangle T is covered if and only if $\bigcap_{g_i \in S_g(T)} C_1(i) \cap T = \emptyset$.*

Proof $\Rightarrow$ If T is covered, a guard is always present at its vertex in case I lies inside it. If I lies inside the critical region associated to a guard, the guard cannot lie on the vertex of T. Therefore, the critical regions associated with the guards cannot have a common intersection inside T since it is covered. $\Leftarrow$ If $\bigcap_{g_i \in S_g(T)} C_1(i) \cap T = \emptyset$, every point inside T lies outside the critical region of at least one $g_i \in S_g(T)$. Therefore, T is covered. □

6 Activation of additional guards

In this section, we present a procedure to activate or deactivate the guards to ensure tracking depending on the instantaneous value of r. Once S_h and S_c are defined and the critical regions of the guards in S^h_g are obtained, we want to determine the number of activated guards based on $\mathbf{v}_I$. First, we introduce some notation. Given $T \in T(G)$, let $V(T)$ and $E(T)$ be the sets of vertices and edges of T respectively. S^{ac}_v, $S^{in}_v \subset S^v_g$ denote the sets of active and inactive vertex guards respectively. $S_v^h \subseteq S_c$ is the set of candidate vertices associated to the diagonals in S_h. Thus, $|S_c| = |S_v^h| + |S_v^{ac}| + |S_v^{in}|$.

Algorithm 2 UpdateActiveGuards

```
1: Input: $P,G,r,S'_T,S_v^h,S_v^{in},S_g^h,S_v^{ac}$
2: Output: Updated $S_v^{ac}$ and $S_v^{in}$
3: if $r$ increased then
4:     for each $g_i \in S_g$ compute $C_1(i)$
5:     $S_T^1 \leftarrow \{T \in S'_T$ : condition of Lemma 2 fails $\}$
6:     while $|S_T^1| > 0$ do
7:         $T_c \leftarrow$ arbitrary $T \in S_T^1$
8:         ActivateGuard($T_c,S'_T,S_v^h,S_v^{in},S_g^h$)
9:         update classification of guards in $S_g^h$
10:        for each $g_i \in S_g^h$ compute $C_1(i)$
11:        $S_T^1 \leftarrow \{T \in S'_T$ : condition of Lemma 2 fails $\}$
12:    end while
13: else if $r$ decreased then
14:    while $S_T^1 = \emptyset$ and $S_v^{ac}! = \emptyset$ do
15:        $g \leftarrow$ arbitrary $g_i \in S_v^{ac}$
16:        deactivate $g$
17:        update classification of guards in $S_g^h$
18:        for each $g_i \in S_g^h$ compute $C_1(i)$
19:        $S_T^1 \leftarrow \{T \in S'_T$ : condition of Lemma 2 fails $\}$
20:    end while
21:    activate $g$
22: end if
```

Algorithm 3 ActivateGuard

```
1: Input: $T_c,S'_T,S_g^h,S_v^h,S_v^{in}$
2: Output: Updated $S'_T$, $S_v^{in}$
3: if $S_v^{in} \cap V(T_c) \neq \emptyset$ then
4:     $v_c \leftarrow$ arbitrary $v \in S_v^{in} \cap V(T_c)$
5:     activate guard in $v_c$
6:     update classification of triangles
7: else
8:     $v_c \leftarrow$ arbitrary $v \in S_v^h \cap V(T_c)$
9:     $g \leftarrow g_i \in S_g^h$ such that $v_c = v_j(i)$
10:     $T_c \leftarrow$ arbitrary $T_c \in T_k(i) \cap S'_T$
11:     ActivateGuard($T_c,S'_T,S_v^h,S_v^{in},S_g^h$)
12: end if
```

We define $S'_T \subset T(G)$ as the set of non-safe triangles of G. Algorithm 2 shows the pseudocode to activate guards based on the condition in Lemma 2. The input to this algorithm is the polygon, the position of the guards, and the instantaneous r. At every iteration, Algorithm 2 searches for a triangle that cannot be covered (Lines 11 and 19) based on Lemma 2. If such a triangle exists, Algorithm 3 searches for a guard in S_v^{in} to activate by the following recursive process. The algorithm starts with a non-safe triangle T_c, if $S_v^{in} \cap V(T_c) = \emptyset$, then it selects a $g_i \in S_g$ followed by a non-safe triangle at the opposite endpoint of h_i. For each non-safe triangle visited by the algorithm we define a vertex a $v \in V(G_{rec})$ and for each g_i used in the algorithm we define an edge $v_1v_2 \in E(G_{rec})$. Thus, a graph G_{rec} is built (refer to Fig. 6b). G_{rec} is a path that ends at a vertex v that corresponds to a triangle T_c where $S_v^{in} \cap V(T_c) \neq \emptyset$.

Lemma 3 *Algorithm 3 terminates in finite steps.*

Proof The recursive procedure chooses a diagonal guard at each step. If the algorithm does not terminate, it implies that either $|S_g^h| = \infty$ (which is not possible), or a cycle exists in G_{rec} (refer to Fig. 6c). The latter cannot occur since the existence of such a cycle implies the existence of Type 3 guards (refer to Fig. 6a). Since all Type 3 guards are converted to Type 1 guards before the algorithm starts (discussed in Sect. 5), G_{rec} cannot have a cycle. □

At the end of each execution of Algorithm 3, at least one non-safe triangle is converted into a safe triangle, and this in turn reduces the area of $\bigcap_{g_i \in S_g(T_c)} C_1(i)$. Moreover, since the number of non-safe triangles is finite, the number of executions of Algorithm 3 is finite. After each iteration of Algorithm 2, a non-safe triangle T gets converted to a safe triangle, and there are no regions inside T that need to be covered by any diagonal guard. This change "relieves" the guards $g_i \in S_g(T)$ of the responsibility of covering T. Consequently, the critical regions $C_1(i)$ decrease their size. Moreover, smaller critical regions $C_1(i)$ increase the chance of the condition of Lemma 2 to be met. Finally, the following lemma shows that Algorithm 2 terminates.

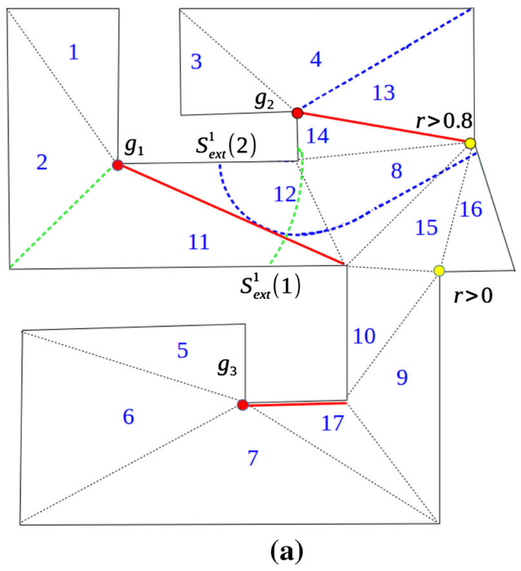
Algorithm 3, forces Lemma 2 to be satisfied for some guards, tracking is guaranteed (maybe this needs to be proved if the paper is self-contained), Lemma 4 proves that Algorithm 2 calls Algorithm 3 (forces Lemma 2 to be satisfied) any time that it is required so tracking is ensured for all the guards, that implies correctness.

Lemma 4 *Algorithm 2 guarantees that the triangle in which the intruder is located is always covered by a guard regardless of* $\mathbf{v}_I$.

Proof The proof is trivial for safe triangles since by definition, they are always covered when there is an intruder inside them according to Lemma 2. For the case of non-safe triangles, Algorithm 2 activates vertex guards for the current value of $\mathbf{v}_I$ until the condition of Lemma 2 is true for all non-safe triangles. Since Algorithm 3 terminates in finite steps, and since after each iteration of Algorithm 3, a non-safe triangle T gets converted to a safe triangle, Algorithm 3 terminates after a finite number of steps. Since the condition of Lemma 2 is met by all the triangles after a finite number of iterations of Algorithm 2, the result follows. □

7 Results

In this section, examples that illustrate the relation between different values of r found through simulation and the number of active guards are presented. Figure 7a shows a simple



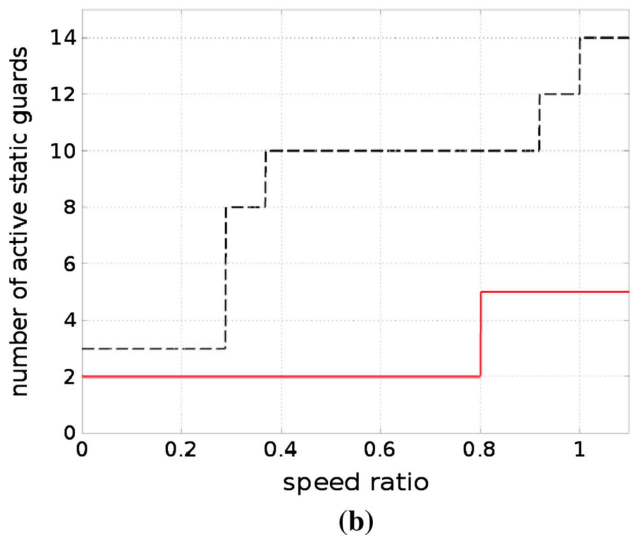


Fig. 7 **a** Each static guard is labeled with the value of r that triggers its activation. **b** The number of activated static guards increases as r grows for examples 1 (red solid line) and 2 (dashed black line) (Color figure online)

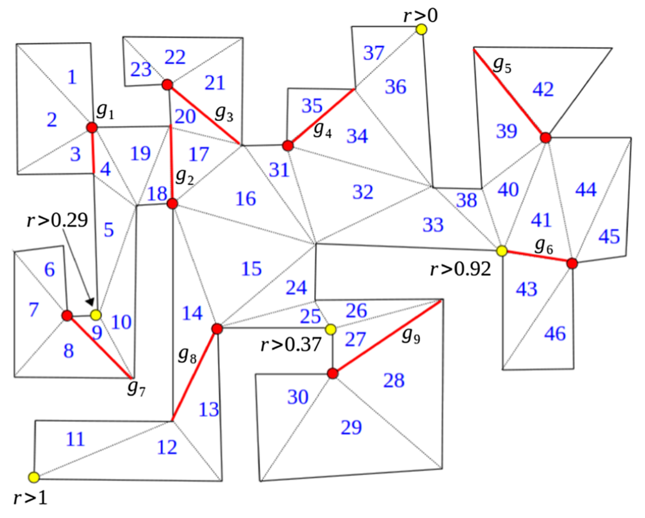


Fig. 8 Each static guard is labeled with the value of r that triggers its activation

polygon with $n = 19$ vertices. The candidate vertices (from Algorithm 1) are marked in red. Diagonal drawn in red represent the diagonal guards. $|S_c| = 5 < \lfloor 19/3 \rfloor = 6$ with $|S^h_g| = 3$, $S^{ac}_v = \emptyset$ and $|S^{in}_v| = 2$. g_3 needs to cover triangles 7 and 9 from each endpoint. When $r > 0$, the critical region of g_3 does not meet the condition of Lemma 2 in triangle 9. As a result, Algorithm 2 activates the inactive guard in triangle 9. Activated vertex guards are marked in yellow with the corresponding r that triggers their activation. After activation, triangles 9, 10, 15 and 16 become safe. It follows that g_3 meets the definition of a Type 0 guard covering triangles 5, 6, 7 and 17 (so all of them become safe triangles). For $0 < r \leq 0.8$, $|S^h_v \cup S^{ac}_v| = 4$. For $r > 0.8$, the condition of Lemma 2 is violated in triangle 8 (the critical regions of g_1 and g_2 intersect in 8). Hence, Algorithm 2 finds the inactive guard in triangle 8 and activates it, thereby, covering triangles 13, 14, 15, 16 and 8. Consequently, g_1 and g_2 become Type 0 covering the remaining triangles. The red plot in Fig. 7b shows the variation in the number of active static guards as r increases. For any $r > 0.8$, the total number of guards does not change since the environment is completely covered at that point. Persistent area coverage is achieved for any value of r greater than 0.8 which implies that even if the intruder has infinite speed the set of 3 diagonal guards and 2 vertex guards suffices to keep track of I.

In Fig. 8, a simple polygon with $n = 48$ vertices representing the environment is illustrated. $|S_c| = 14 < \lfloor 48/3 \rfloor = 16$ with $|S^h_g| = 9$, $S^{ac}_v = \emptyset$ and $|S^{in}_v| = 5$. Since g_5 needs to cover triangles 39 and 42 from both endpoints, it meets the definition of a Type 0 guard so it can cover triangles 39, 42, 40, 41, 44. These triangles along with other triangles with a diagonal guard as an edge are safe triangles. g_4 needs to cover triangles 60 and 76 from each endpoint. When $r > 0$, the critical region of g_4 is such that it causes the condition of Lemma 2 to be violated in triangle 60. Therefore, the inactive guard in triangle 36 becomes active. After the activation, triangles 37 and 36 become safe. It follows that g_4 meets the definition of a Type 0 guard covering triangles 31 and 32.

For $0 < r \leq 0.29$, $|S_g \cup S^{ac}_v| = 10$. When $r > 0.29$, the condition of Lemma 2 is not met in triangle 10. Hence, Algorithm 2 finds the inactive guard in triangle 10 and activates it, so it covers triangles 5, 10 and 9. Consequently, g_7 and g_1 become Type 0. Therefore, g_7 covers triangles 6, 7, 8 and 9. g_1 covers triangles 1, 2, 3, 4 and 19, which causes g_2 to also become Type 0 covering triangles 14, 15, 16, 17 and 18. For $0.29 < r \leq 0.37$, $|S_g \cup S^{ac}_v| = 11$. Nevertheless, when $r > 0.37$, the condition of Lemma 2 is not met in triangle 26. Hence, Algorithm 2 activates the inactive guard located at a vertex of triangle 26, so it covers triangles 25, 26 and 27. g_9 becomes Type 0 so it covers triangles 27, 28, 29 and 30. Following the same procedure, the inactive guard located at triangle 40 is activated when $r > 0.92$. When $r > 1$, the condition of Lemma 2 is violated in triangle 24. Algorithm 2 does not find an inactive guard so it uses g_8 to find a triangle where there is an inactive guard that can be activated. The inactive guard in triangle 12 is activated, and consequently, g_8 becomes a Type 0 covering triangles 13, 14, 15, 24 and 25. The dashed black plot in Fig. 7b shows the variation in the number of active static guards as r increases. For any $r > 1$, the total number of guards does not change since the environment is completely covered at that point. As it has been shown, the number of activated guards depends on r, so for low speed regimes of the intruder, not all guards need to be activated, in contrast with the case of only static guards where the total number of activated guards for both examples would be 5 and 14 respectively regardless of r.

8 Conclusions

In this paper, we explored a problem in which a team of static and mobile guards track a mobile intruder with unknown maximum speed. We presented an algorithm to identify *candidate vertices* in a polygon that can serve as endpoints of diagonal guards or as a seat for vertex guards. We presented an activation strategy for the guards that is adaptive to the instantaneous speed of the intruder. Simulation results for two different environments illustrated the performance of the proposed techniques.

As a part of the future work, the problem of selecting a triangulation that leads to fewer active guards can be considered. Another direction of future work is to consider the case when diagonal guards can be activated/deactivated in order to further reduce the energy consumption of the entire network. Finally, exploring the effect of coordination among the intruders, and limited sensing and motion capabilities of the guards on the tracking performance are relevant problems for future work.

Acknowledgements This work was supported by the NSF Grant IIS-1816343.

References

Aghajan, H., & Cavallaro, A. (2009). *Multi-camera networks: Principles and applications*. Cambridge: Academic Press.

Baraniuk, R. G. (2011). More is less: Signal processing and the data deluge. *Science*, *331*(6018), 717–719.

Başar, T., & Bernhard, P. (2008). *H-infinity optimal control and related minimax design problems: A dynamic game approach*. New York: Springer.

Berg, M., Cheong, O., Kreveld, M., & Overmars, M. (2008). *Computational geometry: Algorithms and applications* (3rd ed.). Santa Clara, CA: Springer.

Bhattacharya, S., Baar, T., & Falcone, M. (2014). Numerical approximation for a visibility based pursuit-evasion game. In *Proceedings of IEEE/RSJ international conference on intelligent robots and systems* (pp. 68–75).

Bhattacharya, S., & Hutchinson, S. (2010). On the existence of Nash equilibrium for a two-player pursuit-evasion game with visibility constraints. *The International Journal of Robotics Research*, *29*(7), 831–839.

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

Bhatti, S., & Xu, J. (2009). Survey of target tracking protocols using wireless sensor network. In *Proceedings of fifth international conference on wireless and mobile communications* (pp. 110–115).

Chen, Y., Zhao, Q., Krishnamurthy, V., & Djonin, D. (2007). Transmission scheduling for optimizing sensor network lifetime: A stochastic shortest path approach. *IEEE Transactions on Signal Processing*, *55*(5), 2294–2309.

Chvatal, V. (1975). A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory Series B*, *18*, 39–41.

Deng, J., Han, Y. S., Heinzelman, W. B., & Varshney, P. K. (2005). Scheduling sleeping nodes in high density cluster-based sensor networks. *Mobile Networks and Applications*, *10*(6), 825–835.

Durocher, S., & Mehrabi, S. (2013). Guarding orthogonal art galleries using sliding cameras: Algorithmic and hardness results. In K. Chatterjee & J. Sgall (Eds.), *Mathematical foundations of computer science 2013* (pp. 314–324). Berlin: Springer.

Hausman, K., Müller, J., Hariharan, A., Ayanian, N., & Sukhatme, G. S. (2015). Cooperative multi-robot control for target tracking with onboard sensing. *The International Journal of Robotics Research*, *34*(13), 1660–1677.

He, Y., & Chong, E. K. (2006). Sensor scheduling for target tracking: A Monte Carlo sampling approach. *Digital Signal Processing*, *16*(5), 533–545.

Hoffmann, F. (1990). On the rectilinear art gallery problem. In M. S. Paterson (Ed.), *Automata, languages and programming: Proceedings of 17th international Colloquium Warwick University, England* (pp. 717–728). Berlin: Springer.

Ho, Y., & Lee, R. (1964). A Bayesian approach to problems in stochastic estimation and control. *IEEE Transactions on Automatic Control*, *9*(4), 333–339.

Howard, A., Mataric, M. J., & Sukhatme, G. S. (2002). Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. *Distributed Autonomous Robotic Systems*, *5*, 299–308.

Isler, V., Kannan, S., & Khanna, S. (2004). Randomized pursuit-evasion with limited visibility. In *Proceedings of ACM-SIAM symposium on discrete algorithms (SODA)* (pp. 1053–1063).

Jung, B., & Sukhatme, G. S. (2002). Tracking anonymous targets using a robotics sensor network. In *AAAI spring symposium*. AAAI Press.

Kumar, S., Lai, T. H., & Balogh, J. (2004). On k-coverage in a mostly sleeping sensor network. In *Proceedings of the 10th annual international conference on mobile computing and networking*, MobiCom '04 (pp. 144–158). New York, NY: ACM.

Laguna, G. J., & Bhattacharya, S. (2017). Hybrid system for target tracking in triangulation graphs. In *Proceedings of IEEE international conference on robotics and automation (ICRA)* (pp. 839–844).

Laguna, G. J., Zou, R., & Bhattacharya, S. (2016). Target tracking on triangulation graphs. In *Proceedings of IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 460–465).

LaValle, S., Lin, D., Guibas, L., Latombe, J. C., & Motwani, R. (1997). Finding an unpredictable target in a workspace with obstacles.

In *Proceedings of IEEE international conference on robotics and automation* (vol. 1, pp. 737–742).

Luke, S., Sullivan, K., Panait, L., & Balan, G. (2005). Tunably decentralized algorithms for cooperative target observation. In *Proceedings of the fourth international joint conference on autonomous agents and multiagent systems*, AAMAS '05 (pp. 911–917). New York, NY: ACM.

Megiddo, N., Hakimi, S. L., Garey, M. R., Johnson, D. S., & Papadimitriou, C. H. (1981). The complexity of searching a graph. In *22nd Annual symposium on foundations of computer science (SFCS 1981)* (pp. 376–385).

Mehmetcik, E., & Ozguner, U. (2013). Centralized target tracking with propagation delayed measurements. In *Proceedings of the 16th international conference on information fusion* (pp. 820–826).

Murrieta-Cid, R., Gonzalez, H., & Tovar, B. (2002). A reactive motion planner to maintain visibility of unpredictable targets. In *Proceedings of IEEE international conference on robotics and automation* (pp. 4242–4248).

O'Rourke, J. (1983). Galleries need fewer mobile guards a variation on Chvátal's theorem. *Geometriae Dedicata*, *14*(3), 273–283.

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

Simon, G., Maróti, M., Lédeczi, A., Balogh, G., Kusy, B., Nádas, A., Pap, G., Sallai, J., & Frampton, K. (2004). Sensor network-based countersniper system. In *Proceedings of the 2nd international conference on embedded networked sensor systems*, SenSys '04 (pp. 1–12). New York, NY: ACM.

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

Theodoridis, T., & Hu, H. (2012). Toward intelligent security robots: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*(6), 1219–1230.

Tian, D., & Georganas, N. D. (2002). A coverage-preserving node scheduling scheme for large wireless sensor networks. In *Proceedings of the 1st ACM international workshop on wireless sensor networks and applications*, WSNA '02 (pp. 32–41). New York, NY: ACM.

Werner-Allen, G., Lorincz, K., Ruiz, M., Marcillo, O., Johnson, J., Lees, J., et al. (2006). Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, *10*(2), 18–25.

Williams, R. K., & Sukhatme, G. S. (2015). Observability in topology-constrained multi-robot target tracking. In *Proceedings of IEEE international conference on robotics and automation (ICRA)* (pp. 1795–1801).

Zou, R., & Bhattacharya, S. (2015). Visibility based multi-agent surveillance strategies in decentralized network. In *SPIE Defense + Security* (pp. 94,640P–94,640P). International Society for Optics and Photonics.

Guillermo J. Laguna received his B.Eng. degree in Mechatronics Engineering from University of Guanajuato, México in 2011. He received his MS in Computer Science from the Centro de Investigación en Matemáticas (CIMAT), México in 2013. He is currently pursuing a Ph.D. degree in Mechanical Engineering at Iowa State University. His research interests are in robotics, especially exploration of unknown environments, target tracking and computational geometry.

Sourabh Bhattacharya is an Assistant Professor at Iowa State University affiliated to the department of Mechanical Engineering and Computer Science. He received his MS in Electrical Engineering, MA in Applied Mathematics, and Ph.D. in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign in 2010. His research interests are in game theory and motion planning for multirobot systems.