2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)
December 4-5, 2023. Boston, USA

From Distributed Coverage to Multi-agent Target Tracking

Shashwata Mandal[1] and Sourabh Bhattacharya[1,2]

2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS) | 979-8-3503-7076-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/MRS60187.2023.10416773

***Abstract*— In this work, we address the problem of deploying mobile agents that try to visually track a mobile intruder in a polygonal environment. First, we present an algorithm to generate a set of regions (referred to as *tiles*) that can ensure distributed coverage of the entire environment. Next, we propose a tracking strategy for a line guard to track a mobile intruder. Finally, we propose resource-constrained deployment strategies for the team. Simulation results are presented to demonstrate the efficacy of the proposed technique.**

I. INTRODUCTION

Advancements in hardware and algorithmic prowess of modern robotic systems has provided effective solutions to real-world problems [1]. In the recent years, surveillance technology has made significant strides [2] in harnessing the power of autonomous multi-robot systems. Building on these technological feats, the intricate domain of target-tracking has gained prominence [3]. Target-tracking refers to the problem of planning the motion of a mobile observer that tries to track a mobile intruder in the presence of obstacles [4]. In this work, we use tools from computational geometry to find a set of salient points for mobile guards in a simply-connected environments that ensure visual coverage.

Non-convex environments present unique challenges in visibility, creating blindspots that can impede accurate tracking [5]. These blindspots [6] (or gaps or shadow regions), essentially lapses in visibility, accentuate the need for a solution rooted in algorithmic sophistication and geometric understanding. Motivated from computational geometry, our approach re-envisions the traditional understanding of the kernels of polygons [7][8], proposing a tailored solution for these geometrically complex spaces. While most prevailing techniques for visual coverage are based on polygon partitioning (triangulation and its extensions) [9][10], we advocate for a fresh "kernel-centric" approach. This emphasis on the kernel as the core of our methodology allows for an optimized guard deployment strategy, catering to both coverage and effective tracking. By recentering the focus in this manner, we transform the way challenges presented by polygons are addressed, ensuring a more holistic and efficient surveillance technique.

Tracking is closely related to the problem of coverage [11]. Given a set of points that cover the polygon, mobile sensors can be deployed to travel between the points to prevent an intruder from breaking the line-of-sight [12]. The Art Gallery problem and its variants [9] propose deployment strategies for multiple guards to visually cover an environment. In [13], authors show that $\lfloor \frac{n}{3} \rfloor$ point guards are always sufficient and sometimes necessary to cover a n-vertex polygon. For special cases, this bound can be further reduced (for example, $\lfloor n/4 \rfloor$ point guards are sufficient, and in some cases necessary, to cover the interior of an orthogonal polygon with n vertices [14][15][9]). Mobility further reduces the aforementioned bounds. For example, $\lfloor n/4 \rfloor$ diagonal guards are sufficient to cover a general polygon [9]. [16] provides an upper bound of $\lfloor \frac{3n+4}{16} \rfloor$ for mobile guards to cover any n-vertex orthogonal polygon. Based on techniques on polygon partitioning [17] and convex optimization [18], authors in [19] propose a strategy and maximum speed for $\Omega(\frac{n}{6})$ guards to track a mobile intruder (maximum speed known) in a simply-connected environment. The dichotomy between mobile guards and point guards has consistently been deliberated as a trade-off between utility and cost - while the latter necessitates more hardware, the former entails lower operational costs [20].

In the past, researchers have investigated target-tracking in the framework of mathematical optimization. [21] provides a technique to compute motion for a mobile observer that tracks a predictable target in obstacle-rich environments. For unpredictable targets, persistent tracking can be formulated as a stochastic control problem if the behaviour of the target is assumed to be random [22]. For a strategic target that is adversarial, game theory has been used to obtain optimal strategies for the observer in simple environments [5] [23][24]. In case of complex environments, advanced tools for hybrid systems [25], sampling-based motion planning [26], and numerical methods [27] have been investigated to plan motion for the observer. [28] presents a hierarchical motion planner to deploy a team of drones for target-tracking applications in practice. Recent efforts [4] that draw connections between the problem of path planning for a mobile observer trying to track a mobile intruder, and the well-known watchman-route problem [29][30] provide some guarantees on the tracking performance.

The contributions of this work are as follows. (i) We present a novel technique to compute regions inside the polygon that can provide decentralized coverage for arbitrary polygons. (ii) We present control strategy for a line guard (guarding multiple tiles) to track a mobile intruder. (iii) We propose deployment strategy for a mixed team of static and mobile guards to track an intruder that takes into account constraints on team size and guard speed.

The paper is organized as follows. Section II presents the problem formulation. Section III presents a generalization of kernel to arbitrary polygons, and the resulting partitioning of the polygon. Section IV presents a clustering based approach to design strategy for a line guard to track an intruder. Section

[1]Department of Computer Science, Iowa State University, Ames, IA 50010, USA, [2]Department of Mechanical Engineering, Ames, IA 50010, USA smandal@iastate.edu, sbhattac@iastate.edu

V presents deployment strategies that take into consideration resource constraints to optimize team-based metrics. Section VI presents simulation and experimental results. Section VII presents our conclusions and future work.

II. Problem Formulation

Consider a simply-connected polygonal environment with n vertices containing a mobile *intruder*. A team of mobile agents, called *guards*, is deployed in the environment to track the intruder. We assume that all the mobile agents are holonomic, and have a bounded speed. Additionally, the guards have an omni-directional field-of-view without any limitations on the range of visibility. In other words, any point in the polygon that can be connected to the agent by a straight line in free space is visible to it. We assume that the guards and the intruder have a maximum speed denoted by v_g and v_e, respectively and that the team of guards can communicate among themselves. Additionally, it is assumed that the initial position of the intruder is known to the guards. Given the aforementioned scenario, we address the following: (1) Design an algorithm to find a set of points inside the polygon that covers it? (2) For mobile guards, find a deployment strategy that visits the aforementioned points to track an unpredictable mobile intruder inside the environment? From the art-gallery problems, we know that $\lceil \frac{n}{3} \rceil$ static guards are sufficient and sometimes necessary to cover a simply-connected polygonal environment, where n is the number of vertices of the polygon [13][31][14]. Therefore, $\lceil \frac{n}{3} \rceil$ is also a trivial upper bound for tracking. This is a trivial upper bound for tracking. Sections V and VI demonstrate that this bound can be further reduced using the technique proposed in Sections II and III when the guards are mobile.

III. Deployment for distributed coverage

Consider a simply-connected polygon P with n vertices labelled $\{v_0, v_1, \ldots, v_{n-1}\}$. Two points within P, denoted p and q, are mutually visible if the line segment joining them doesn't intersect ∂P (the boundary of P). A vertex v_i is termed a *reflex vertex* (or equivalently, a *corner*) if its internal angle exceeds 180°. An edge in P is defined as a *reflex edge* if it is incident to a reflex vertex. For a reflex vertex v_i, its associated *star region*, represented by $SR(v_i)$, is the region within P bounded by the extension of reflex edges adjacent to v_i and visible from v_i. The kernel of P [32][33] ($\mathrm{Ker}(P)$) is formally defined as follows:

$$\mathrm{Ker}(P) = \bigcap_{v_i \text{ is reflex}} H(v_i)$$

where $H(v_i)$ is the half-plane at v_i which is an extension of the reflex edges contained inside the polygon. A guard located at any point belonging to $\mathrm{Ker}(P)$ can see the entire polygon. However, polygons can have an empty kernel in which case a single guard cannot cover the entire polygon.

Algorithm 1 presents a technique to find a set of regions inside the polygon that provide distributed visual coverage of the entire polygon. Algorithm 1 iteratively selects a reflex vertex, denoted as u for a given polygon. For vertices sequenced as $u, \ldots, w, \ldots, v$, where v represents the ensuing reflex vertex in an anti-clockwise traversal and w is the terminal non-reflex vertex seen from u. Line 6 intersects the star region of u in Q with w's visibility polygon, modifying the star region while Line 7 updates the polygon by omitting vertices between u and w and introduces edge uw. When a reflex vertex u is chosen, any point within the convex region $u, \ldots, w$ remains visible to the modified star region at u. If u is selected later, the visibility holds since a post-iteration modified star region at a reflex vertex is always a subset of its pre-iteration counterpart. Consequently, regions pruned in interim steps are fully visible from all locations in the ending modified star region at the reflex vertex. The process concludes once the polygon becomes convex or retains a single reflex vertex.

The overall time complexity of Algorithm 1 is $O(n^4 \log n)$, where n is the number of vertices of the polygon[1]. The output of Algorithm 1 comprises a collection of disjoint non-empty regions $\{R_i\}_{i=1}^{k}$ (k is called the *cardinality*) such that $\bigcup_{i=1}^{k} VP(p_i) = P$ for any $p_i \in R_i$.

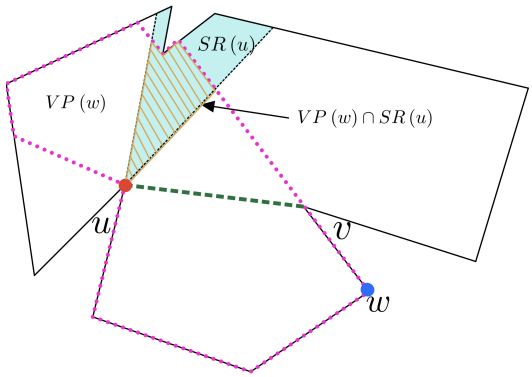


Fig. 1: The shaded region shows the intersection between the star region of u ($SR(u)$), and the visibility polygon of w ($VP(w)$).

Lemma 1. *The output of Algorithm 1 is the same as the intersection of all the star regions of the polygon if the intersection is non-empty.*

Proof. Let S represent the non-empty intersection of all the star regions within the polygon. Suppose S' is the output of Algorithm 1 for this polygon. Given the way the modified star regions are constructed in Algorithm 1, it's clear that $S' \subseteq S$. Now, suppose there exists a point p such that $p \in S$ but $p \notin S'$. This implies that a portion of the polygon isn't visible from p, contradicting the fact that p is part of every star region (since $p \in S$). This contradiction indicates that no such point p can exist, ensuring every point in S is also contained in S'. ■

[1] visibility polygon computations at $O(n \log n)$ [34] and polygon intersections at $O(n^2)$ [8]

Algorithm 1 Star Region Generator

Input: Q Polygon, C corners
Output: SR Set of star regions

function STAR GEN(Q,C)
  $SR(j) \leftarrow VP(j) \cap EdgeExtension(j), \forall j \in C$
  $Q' \leftarrow Q, C' \leftarrow C$
  while $|C'| > 1$ **do**
    Find a corner u such that v is the next reflex vertex and w is the last non-reflex vertex visible to u.
    $SR(c) \leftarrow SR(c) \cap VP(w)$
    Remove all vertices from Q' starting from u till w excluding u and w
    if v is visible to u **then**
      Remove w from Q'
    end if
    Remove $u \in C'$, if u is not a corner in Q'
  end while
  return SR
end function

Given a tile R_i with vertex set $v_1, \ldots, v_n$, we can define its visibility polygon $VP(R_i) = \cap_{j \in \{1,\ldots,n\}} VP(v_j)$. In other words, the visibility polygon of a tile consists of the regions inside the polygon that are visible from all points inside the tile. Tiles R_i and R_j are said to be *mutually visible* if a line-of-sight exists between any two points $p_i \in R_i$ and $p_j \in R_j$.

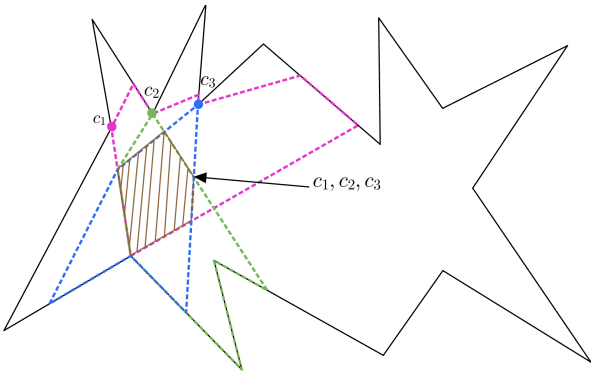
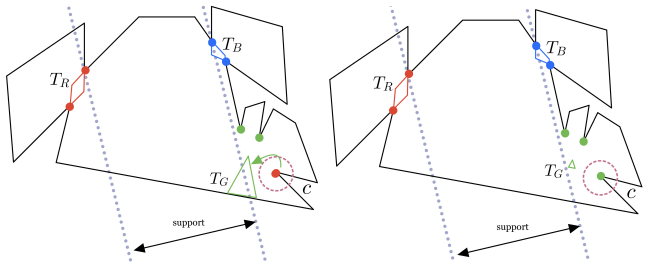


Fig. 2: The shaded region shows the tile generated by the intersection of the star regions from the reflex vertices c_1, c_2 and c_3. The star regions at the corners are generated by Algorithm 1.

IV. TRACKING STRATEGY FOR A LINE GUARD

If the number of guards is equal to the number of tiles, a guard deployment in each tile covers the entire polygon. From hereon, we will refer to guards which are confined to a tile as a static guard. In this section, we address the scenario in which the number of guards is strictly less than the number of tiles. In such cases, some guards have to be responsible for two or more tiles. In this work, we specifically address the case when a guard is deployed to track the intruder inside the union of the visibility polygon of the two tiles. At first, we explain the concept of corner segregation and supporting hyper-planes to locally re-assign corners between two tiles that are mutually visible.

Our proposed control strategy relies on parallel separation planes between the corners of two mutually visible tiles. For planar environments, the separation hyperplanes are parallel lines (called the *support vectors*) that divide the plane into three regions - two half-planes and a section of the plane bounded by the support vectors. Each half-plane contains a tile and the corners associated with it. Compound tiles are generated by intersecting local overlapping tiles using heuristics on local geometry. Support vector machines (SVM) is a learning algorithm used to find a hyperplane between multi-dimensional data to segregate points [35]. We use SVM on the corners associated with a pair of mutually visible tiles to generate the line of segregation. We use SVM in two dimensions to find corners in tiles that are outliers from their **locality** [36]. To segregate a corner from its locality, we use a linear kernel on a standard support vector machine and apply it to two adjacent tiles. Using the line of segregation, we find corner outliers in the tiles. A corner outlier is a corner in a tile that lies on the half plane for the other tile. Once we identify corner outliers, we try to assign them to existing tiles. If such a tile doesn't exist, we generate a separate tile for the corner using the star region for that corner. This ensures that the locality information of the cluster of corners is preserved in the tile allowing us to move to the next step. Figure 3a demonstrates this in detail. When SVM is applied the corners are segregated by support vectors. Corner which lie on the wrong side of the segregation are considered outliers.



(a) (b)

Fig. 3: Figure a) describes the tiles before applying SVM to the corners in the red tile and the blue tile. Initially corner c is a member of the red tile. Figure b) describes the tiles after reassignment. We can see that corner c has become an outlier for the red tile; hence it is now reassigned to the green tile.

Figure 4 shows two mutually visible tiles T_1 (constituent corners $a_1, \ldots, a_n$) and T_2 (constituent corners $b_1, \ldots, b_m$) in the tiles of the polygon. The guard moves on the shortest line segment H connecting the two tiles. Let g_1 and g_2 denote the points at which the segment is incident on tiles T_1 and T_2, respectively.

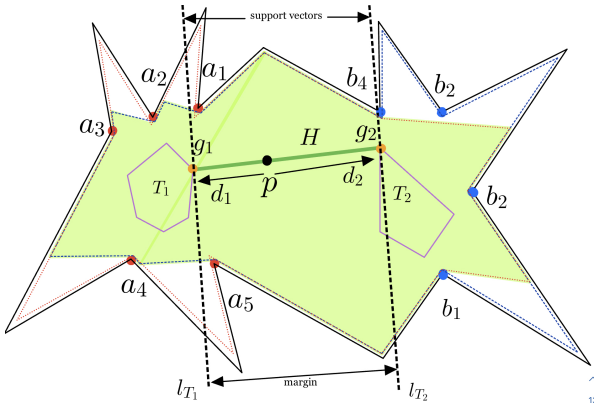


Fig. 4: Figure shows a polygon and its associated tiles. The corners $a_1,\ldots,a_5$ are associated to T_1 and $b_1,\ldots,b_4$ are associated to T_2. The green region is the intersection of the visibility polygons of T_1 and T_2.

A necessary condition for the guard to persistently track the intruder is to prevent it from reaching any corner before the guard reaches the tile associated with the corner [5]. Therefore the following equation must hold for all corners a_i and b_i:

$$\alpha\|x_e - x_{a_i}\|_2 \geq d_1, \quad \alpha\|x_e - x_{b_i}\|_2 \geq d_2, \tag{1}$$

where $\alpha = \frac{v_g}{v_e}$, x_e is the position of the intruder, x_{a_i} and x_{b_i} are the positions of corners a_i and b_i, respectively. Please note that the $\|\|_2$ in (1) is replaced by the minimum distance of the intruder to the corner on the visibility graph [2] (also called the *geodesic distance* [38]) if the corner is not directly visible to the intruder. If (1) holds for the corners a_i and b_i closest to the intruder, then it holds for all the corners in the environment. At each point inside the polygon, there is a pair (a_i,b_i) closest to it.

Algorithm 2 Voronoi Segmentation

Input: Q Polygon, $T = \{T_1,T_2\}$ Tiles
Output: V Vornoi Diagram with extremal points

1: **function** EXTEMEAL VORONOI(Q,T)
2: $C \leftarrow T_1 \cap T_2$ where T_1,T_2 are a set of corners
3: $V \leftarrow GeodesicVoronoi(P,C)$
4: $V_1 \leftarrow GeodesicVoronoi(P,T_1)$
5: $V_2 \leftarrow GeodesicVoronoi(P,T_2)$
6: Intersect each cell associated with corners in T_1 in V with T_2
7: Repeat previous step for T_2 and V_1
8: **return** V
9: **end function**

Algorithm 2 partitions any environment into regions based on pair of corners (a_i,b_i) closest to each point in the environment. Figure 5 presents the output of Algorithm 2 for the polygon in figure 4.

[2]Visibility graph is a graph of mutually visible locations inside a polygon [37].

Given any position of the intruder, the guard should be able to ensure that it can reach both tiles before the intruder can reach the nearest corner constituting the tiles. This implies the following:

$$x_g = L\frac{\|x_e - x_{a_i}\|_2}{\|x_e - x_{a_i}\|_2 + \|x_e - x_{b_i}\|_2}, \tag{2}$$

where x_g is the position of the guard on H with respect to g_1, and a_i and b_i are corners closest to the intruder. Figure 6 shows a polygon with two tiles. Each cell of the partition has a distinct pair of corners nearest from the two tiles. The curves in each partition are locus of evader positions with a constant value of $\frac{\|x_e - x_{a_i}\|_2}{\|x_e - x_{b_i}\|_2}$, where a_i and b_i are corners closest to the intruder. Each curve is a locus of evader positions for which the pursuer has a fixed position on its path.

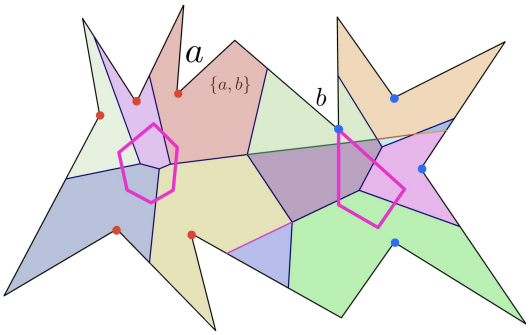


Fig. 5: Figure shows output of Algorithm 2 for the polygon. Each cell has a unique pair (a_i,b_i) nearest to the points in the cell.

Equation (2) leads to the following control law for the guard:

$$v_g = L\frac{\|x_e - x_{a_i}\|_2}{\|x_e - x_{a_i}\|_2 + \|x_e - x_{b_i}\|_2}\vec{v}_e \cdot \left[\overrightarrow{(x_e - x_{a_i})} - \frac{\overrightarrow{(x_e - x_{a_i})} + \overrightarrow{(x_e - x_{b_i})}}{\|x_e - x_{a_i}\|_2 + \|x_e - x_{b_i}\|_2}\right] \tag{3}$$

Equation (1) implies that the following holds for all pairs of corner (a_i,b_i):

$$\begin{aligned}\alpha &\geq \frac{d_1 + d_2}{\|x_e - x_{a_i}\|_2 + \|x_e - x_{b_i}\|_2} \\ &\geq \frac{L}{\min\limits_{a_i,b_i}\|x_{a_i} - x_{b_i}\|_2},\end{aligned} \tag{4}$$

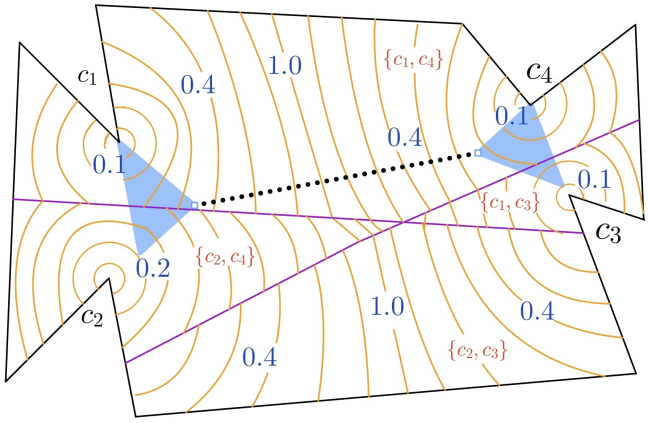


Fig. 6: Figure shows a polygon with 2 tiles (shaded in blue). The level curves for $\frac{\|x_e - x_{a_i}\|_2}{\|x_e - x_{b_i}\|_2}$ are plotted in each partition of the polygon. The violet lines are the boundary of the Voronoi cells associated with the tiles constructed from Algorithm 2.

(4) provides the minimum α necessary for the guard to track the intruder. If a_i and b_i are not mutually visible, then $\|\|_2$ is replaced by the geodesic distance between the two points. The denominator in (4) is lower bounded by the distance between the support vectors obtained by SVM. The distance between the closest corners in T_1 and T_2 increase with the increase in the margin length between the support vectors. **Hence, computing a wider margin facilitates a lower minimum speed requirement for the guard.**

V. Deployment of a Mixed Team of Static and Mobile Guards

In previous section, we presented a strategy for a line guard to cover two mutually visible tiles so that it can track a mobile intruder that lies in the union of the visibility polygon of the tiles. In this section, we address the problem of deploying a team of static and mobile guards on the tiles of an arbitrary polygon. The two parameters that are of interest are the number of guards and the maximum speed of the guards. The aforementioned parameters play an important role in resource-constrained deployment scenarios where the system operator often faces a dilemma between choosing a large number of cheap static sensors or a few costly mobile sensors.

Algorithm 3 Deployment with n guards

Input: K Tiles, $G = (V, E)$ Tile Graph, k guards
Output: D Deployment list

1: **function** DeploymentLimitedGuards(K, G, k)
2: For $e \in E$, calculate α_{max}
3: Store E with α_{max} in a min-priority queue Q
4: $m \leftarrow |K|$, $D \leftarrow \phi$
5: Allocate a pair of tiles in G to a guard till $k < m$ unless $k = 0$ in which case return "Not Possible"
6: Add a static guard to D for remaining tile in K
7: **return** D
8: **end function**

First, we consider the problem of minimizing the speed required by a fixed number of guards to track an intruder inside an arbitrary polygon. Let K denote the cardinality of the tile, and m denote the number of guards. For $m \geq K$, guards can be deployed in each tile for covering the entire polygon. Algorithm 3 addresses the case when $m < K$. It provides a deployment for a team of static and mobile guards that minimizes the speed required for the guards to track the intruder.

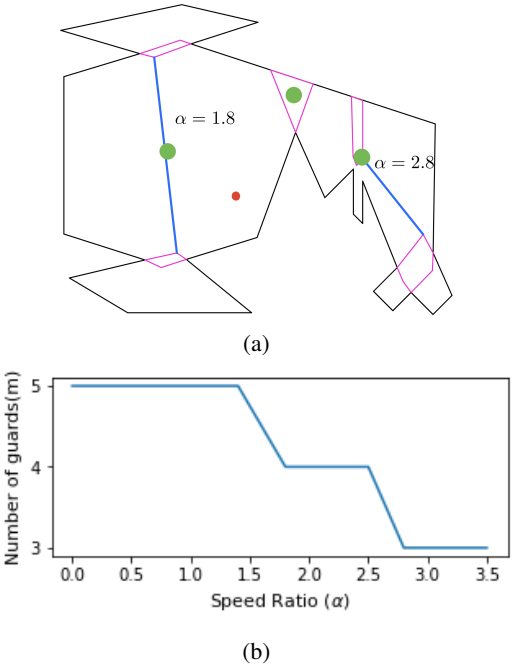


Fig. 7: (a) shows an environment with 5 tiles. The green circles represent the guards (1 static and 2 mobile). (b) shows a plot of the minimum number of guards (m) deployed as a function of the speed ratio α.

Each tile graph edge generates a maximum value of α. In Algorithm 3, Line 3 arranges all edges in an ascending order priority queue. Then we take an edge from the queue and check if a guard has already been assigned to an incident vertex (Line 10). If not, we assign a guard to the edge (i.e., a line guard deployed between tiles) (Line 11). When the number of remaining guards is equal to the number of unassigned tiles, a guard is allocated to each tile. If there are no more guards left to be allocated to unassigned tiles, there is no feasible solution (the algorithm returns "Not possible") (Line 7).

Next, we address the dual problem. Algorithm 4 provides a strategy to deploy minimum number of guards when there is a constraint on their maximum speed. Since each tile graph edge can be used to generate a maximum α, we can do so by limiting the number of edges being considered based on the maximum value of α. The idea is similar to Algorithm 3. The main difference is in Line 3. Contrary to Algorithm 3, Algorithm 4 populates the priority queue with edges with α lower than the input maximum speed.

Algorithm 4 Deployment with max α α_g

Input: K Tiles, G Tile Graph, α_g Max Alpha
Output: D Deployment list, k guards needed

function DEPLOYMENTLIMITEDSPEED(K, G, α_g)
  For $e \in E$, calculate α_{max}
  Store $\forall e \in E, \alpha_{max} \leq \alpha_g$ in a min-priority queue Q α_{max}
  $m \leftarrow |K|$, $k \leftarrow 0, D \leftarrow \phi$
  Allocate a pair of tiles in G to a guard till Q is not empty
  Add a static guard to D for remaining tile in K
  return D, k
end function

Figure 7a shows a polygon with 10 reflex vertices. An intruder is located inside the polygon and its initial position is denoted by the red dot. The polygon contains 5 ($= k$) tiles. The figure shows the line segments on which mobile guards are deployed as the value of α increases along with their initial location. Figure 7b shows a plot of the minimum number of guards (m) required to track the intruder as a function of α. The plot clearly shows the trade-off between m and α for the polygon which is a universal phenomenon is surveillance-related deployments.

VI. SIMULATIONS

In this section, we implement our proposed deployment and tracking strategy for sample environment and present our simulation results.

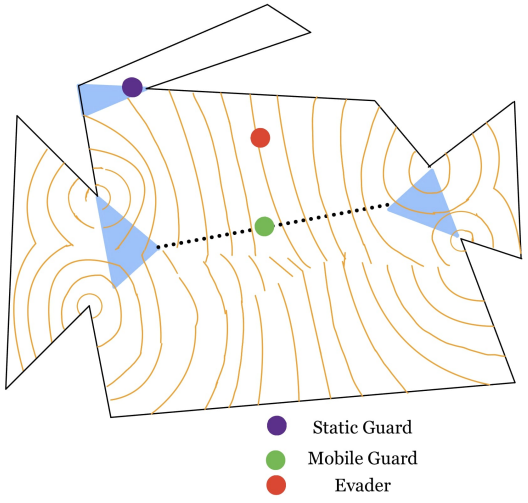


Fig. 8: Figure shows an environment with 3 tiles and 2 guards. The mobile guard and the static guard are shown in green and in purple respectively. The red circle denotes the location of the intruder. The yellow curves in the polygon are the level curves for the voronoi cells associated with tiles covered by the line guard.

Figure 8 shows a polygonal environment with 5 reflex vertices. The initial position of the intruder is shown as a red circle. The polygon has tiles of cardinality 3 as shown in the figure. For $\alpha \geq 2.6$, two tiles can be guarded by a line guard as shown in the figure with the remaining tile being covered by a static guard. For $\alpha \leq 2.6$, a guard needs to be deployed for each tile. The figure shows the locus of intruder positions that leads to a fixed location of the line guard as discussed in Section IV.

In the next simulation, we study the effects of varying the maximum allowable α on the number of guards and their deployment strategy. Figure 9 shows a polygonal environment with 40 vertices of which 16 vertices are reflex. A heuristic algorithm is used to generate tile intersections which reduces the cardinality from 16 to 6. Initially, the maximum α provided for the environment is 2.6. Due to a relatively high value of maximum α, Algorithm 4 generates a deployment of 2 mobile guards and 2 static guards. However, when the value of α is reduced to 2.2, Algorithm 4 generates a deployment of 1 mobile guard and 4 static guards. The edge with $\alpha = 2.6$ gets removed and the green guard is replaced by two yellow guards in the tiles hosting the mobile guard.

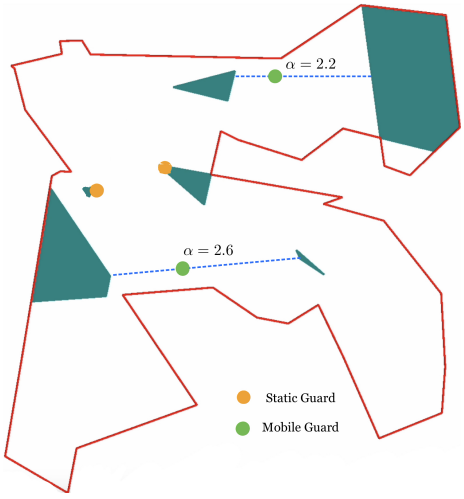


Fig. 9: Figure shows an environment with 40 vertices and 16 corners. The dark cyan polygons highlight the tiles of the polygon. The yellow circles show the location of the static guards and the green circles represents the mobile guards.

VII. CONCLUSION AND FUTURE WORKS

In this work, we addressed the problem of deploying mobile agents that try to visually track a mobile intruder in a polygonal environment. First, we presented an algorithm to generate a set of regions (referred to as *tiles*) that can ensure distributed coverage of the entire environment. Next, we proposed a tracking strategy for a line guard to track a mobile intruder. Finally, we proposed resource-constrained deployment strategies for the team. Simulation results are presented to demonstrate the efficacy of the proposed technique.

In the future, we plan to extend this technique to take into account motion and sensing constraints for the guards. Extension of the technique for tracking in 3-d environments [39][40] is another interesting direction of future research.

REFERENCES

[1] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[2] R. Ayad. (2023) Security trends: Four surveillance trends for 2023 (and beyond). Accessed: 2023-05-25. [Online]. Available: https://www.asisonline.org/publications--resources/news/blog/2023/security-trends-four-surveillance-trends/

[3] M. Kumar and S. Mondal, "Recent developments on target tracking problems: A review," *Ocean Engineering*, vol. 236, p. 109558, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0029801821009471

[4] G. J. Laguna and S. Bhattacharya, "Path planning with incremental roadmap update for visibility-based target tracking," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 1159–1164.

[5] S. Bhattacharya and S. Hutchinson, "A cell decomposition approach to visibility-based pursuit evasion among obstacles," *The International Journal of Robotics Research*, vol. 30, no. 14, pp. 1709–1727, 2011. [Online]. Available: https://doi.org/10.1177/0278364911415885

[6] B. Tovar, L. Guilamo, and S. LaValle, *Gap Navigation Trees: Minimal Representation for Visibility-based Tasks*, 10 2005, vol. 17, pp. 425–440.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009.

[8] M. I. Shamos and D. Hoey, "Geometric intersection problems," in *17th Annual Symposium on Foundations of Computer Science (sfcs 1976)*, 1976, pp. 208–215.

[9] J. O'Rourke, *Art gallery theorems and algorithms / Joseph O'Rourke.*, ser. International series of monographs on computer science ; 3, 1987.

[10] M. Keil, "Polygon decomposition," *Handbook of Computational Geometry*, 01 2000.

[11] Y. Amit, J. Mitchell, and E. Packer, "Locating guards for visibility coverage of polygons." *Int. J. Comput. Geometry Appl.*, vol. 20, pp. 601–630, 01 2010.

[12] A. Ganguli, J. Cortés, and F. Bullo, "Distributed deployment of asynchronous guards in art galleries," *2006 American Control Conference*, pp. 6 pp.–, 2006. [Online]. Available: https://api.semanticscholar.org/CorpusID:13264364

[13] V. Chvátal, "A combinatorial theorem in plane geometry," *Journal of Combinatorial Theory, Series B*, vol. 18, no. 1, pp. 39–41, 1975. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0095895675900611

[14] J. Kahn, M. Klawe, and D. Kleitman, "Traditional galleries require fewer watchmen," *SIAM Journal on Algebraic Discrete Methods*, vol. 4, no. 2, pp. 194–206, 1983. [Online]. Available: https://doi.org/10.1137/0604020

[15] E. Györi, "A short proof of the rectilinear art gallery theorem," *Siam Journal on Algebraic and Discrete Methods*, vol. 7, pp. 452–454, 1986. [Online]. Available: https://api.semanticscholar.org/CorpusID:121732626

[16] A. Aggarwal, "The art gallery theorem: its variations, applications and algorithmic aspects," 1984.

[17] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, 2nd ed. Springer-Verlag, 2000. [Online]. Available: http://www.cs.uu.nl/geobook/

[18] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[19] H. Emadi, T. Gao, and S. Bhattacharya, "Visibility-based target-tracking game: Bounds and tracking strategies," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 1917–1924, 2017.

[20] E. Győri and T. R. Mezei, "Mobile versus point guards," *Discrete &amp Computational Geometry*, vol. 61, no. 2, pp. 421–451, apr 2018. [Online]. Available: https://doi.org/10.1007%2Fs00454-018-9996-x

[21] S. LaValle, H. Gonzalez-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*, vol. 1, 1997, pp. 731–736 vol.1.

[22] G. P. Huang, K. X. Zhou, N. Trawny, and S. I. Roumeliotis, "Bearing-only target tracking using a bank of map estimators," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 4998–5005.

[23] S. Bhattacharya and S. Hutchinson, "On the existence of nash equilibrium for a two-player pursuit—evasion game with visibility constraints," *The International Journal of Robotics Research*, vol. 29, no. 7, pp. 831–839, 2010. [Online]. Available: https://doi.org/10.1177/0278364909354628

[24] R. Zou and S. Bhattacharya, "On optimal pursuit trajectories for visibility-based target-tracking game," *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 449–465, 2019.

[25] G. J. Laguna and S. Bhattacharya, "Hybrid system for target tracking in triangulation graphs," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 839–844.

[26] G. Laguna, S. Mandal, and S. Bhattacharya, "Roadmap for visibility-based target tracking: Iterative construction and motion strategy," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 4732–4737.

[27] S. Bhattacharya and T. Başar, "Spatial approaches to broadband jamming in heterogeneous mobile networks: a game-theoretic approach," *Autonomous Robots*, vol. 31, pp. 367–381, 2011. [Online]. Available: https://api.semanticscholar.org/CorpusID:30214890

[28] S. Mandal, T. Gao, and S. Bhattacharya, "Planning for aerial robot teams for wide-area biometric and phenotypic data collection," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 2586–2591.

[29] J. O'Rourke, "Galleries need fewer mobile guards: A variation on chvátal's theorem," *Geometriae Dedicata*, vol. 14, pp. 273–283, 1983.

[30] M. Ernestus, S. Friedrichs, M. Hemmer, J. Kokemüller, A. Kröller, M. Moeini, and C. Schmidt, "Algorithms for art gallery illumination," *Journal of Global Optimization*, vol. 68, pp. 23–45, 2017.

[31] S. Hengeveld and T. Miltzow, "A practical algorithm with performance guarantees for the art gallery problem," *arXiv.org*, 2022.

[32] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*. Berlin, Heidelberg: Springer-Verlag, 1985.

[33] J. Mark, "Studies on Kernels of Simple Polygons," 2020. [Online]. Available: http://dx.doi.org/10.34917/19412121

[34] T. Asano, "An efficient algorithm for finding the visibility polygon for a polygonal region with holes," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 68, pp. 557–559, 1985. [Online]. Available: https://api.semanticscholar.org/CorpusID:119415599

[35] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, p. 273–297, sep 1995. [Online]. Available: https://doi.org/10.1023/A:1022627411411

[36] A. Ben-Hur, "Support vector clustering," *Scholarpedia*, vol. 3, no. 6, p. 5187, 2008, revision #186055.

[37] S. K. Ghosh and D. M. Mount, "An output sensitive algorithm for computing visibility graphs," in *FOCS*, 1987.

[38] C. Liu, "A nearly optimal algorithm for the geodesic voronoi diagram in a simple polygon," *CoRR*, vol. abs/1803.03526, 2018. [Online]. Available: http://arxiv.org/abs/1803.03526

[39] E. Lipka, "A note on minimal art galleries," 09 2019.

[40] S. Lunz, Y. Li, A. W. Fitzgibbon, and N. Kushman, "Inverse graphics GAN: learning to generate 3d shapes from unstructured 2d data," *CoRR*, vol. abs/2002.12674, 2020. [Online]. Available: https://arxiv.org/abs/2002.12674