2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Relay Pursuit for Multirobot Target Tracking on Tile Graphs

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

Abstract— **In this work, we address a visbility-based target tracking problem in a polygonal environment in which a group of mobile observers try to maintain a line-of-sight with a mobile intruder. We build a bridge between data mining and visibility-based tracking using a novel tiling scheme for the polygon. First, we propose a tracking strategy for a team of guards located on the tiles to dynamically track an intruder when complete coverage of the polygon cannot be ensured. Next, we propose a novel variant of the Voronoi Diagram to construct navigation strategies for a team of co-located guards to track an intruder from any initial position in the environment. We present empirical analysis to illustrate the efficacy of the proposed tiling scheme. Simulations and testbed demonstrations are present in a video attachment.**

I. INTRODUCTION

Surveillance systems are an essential part of securing public and private spaces. These systems can range from wireless home security cameras to sophisticated alarm systems that notify law enforcement at the first sign of trouble. Availability of affordable commercially off-the-shelf mobile sensors has catapulted surveillance technology to a new level. Target tracking is a critical component of modern surveillance systems [1][2][3]. Target tracking refers to the problem of designing motion strategies for a mobile observer that tries to track a mobile intruder in an environment containing obstacles. In [4], we proposed an algorithm to construct a *paparazzi route*(an analogue of the *watchman's route* [5][6][7]) that tries to track an intruder in an environment. In this work, we present a novel framework (inspired by data mining techniques) to design tracking strategies for a multirobot system to track an unpredictable target.

Data mining is a prominent tool used for searching and extracting knowledge rules and association rules from large datasets. For example, Associative Rule Mining [8][9] is a prevalent data mining technique used to find association rules between features. We utilize this search functionality used in associative rule mining to find visibility associations between spatial tessellations of a polygon. Spatial tessellations have recently gained a lot of traction due to their ability to encode geometric information eg. Penrose Tiling [10].

Traditionally, visibility-based coverage problems entail partitioning the environment into convex regions to place the guards [11]. A widely used partitioning technique is *triangulation* [12] which involves covering any planar polygon using triangles. A key geometric feature of the environment that plays a pivotal role in visibility-based surveillance problems is a reflex vertex [13]. Reflex vertices (corners) induce discontinuity in the sensing footprint of a camera sensor. Maintaining a line-of-sight (LOS) with a mobile entity involves preventing unwanted events around the reflex vertices of the polygon. In this work, we exploit the geometric features of a corner to induce a tiling of the free space of the polygon appropriate for tracking problems.

In this work, we extend results from our previous work around a corner [14] to propose voronoi-based partitioning techniques for evaluating the risk of escape for the intruder in an environment containing multiple corners. In contradistinction to the previous work, our current work relies on the construction of a novel variant of the Voronoi Diagram [15][16] suitable for tracking applications. Specifically, the contributions of our work are as follows. (i) We build a bridge between data mining and visibility-based tracking using a novel tiling scheme of the polygon. (ii) We propose a tracking strategy for a team of guards located on the tiles to dynamically track an intruder when complete coverage cannot be ensured. (iii) We propose a novel variant of the Voronoi Diagram to construct navigation strategies for a team of co-located guards to track an intruder from any initial position in the environment.

The paper is organized as follows. Section II provides the problem formulation. Section III provides an introduction to tiling based on corners of a polygon. Section IV presents an algorithm using data mining techniques to construct a tiling efficiently. Section V presents a strategy for guards placed on the tiles to track an intruder. Section VI presents an algorithm for a team of co-located guards to track an intruder. Section VII presents our simulation results. Section VIII 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. We assume that the team of guards can communicate among themselves. Additionally, it is assumed that the intruder is initially visible to the guards. Given the aforementioned scenario, we ask

*This work was in part supported by USDA/NIFA National Robotics Initiative under Grant 2017-67021-25965 and NSF IIS award 1816343.

[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

the following question: What is a strategy for the guards to move in order to ensure that the intruder is visible to at least one of the guards at any time? It is well-known that $\lceil \frac{n}{3} \rceil$ static guards can cover an entire simply-connected polygonal environment, where n is the number of vertices of the polygon [17][18][19]. However, fewer mobile guards can track an intruder without covering the environment. In [4][20], we proposed tracking strategies for a single mobile guard to track an intruder in a polygonal environment. In our current work, we investigate the target-tracking problem for multiple guards in a polygonal environment.

In the next section, we propose a decomposition of the polygonal environment that provides a manifold for the guards to guarantee persistent tracking (at least one guard in the team can see the intruder at any time instant) for an infinite horizon.

III. Tiling in Polygonal Environments

In this section, we introduce a novel tiling scheme that encodes the visibility of the environment. Consider the polygonal environment shown in Figure 1. A *reflex vertex* is defined as a vertex whose external angle is less than 180°. In Figure 1, c_1, c_2, c_3 and c_4 are the reflex vertices of the polygon. From here on in, we shall use the terms "corner" and reflex vertices interchangeably for the rest of the paper. A *star region* associated with a corner is defined as a modified kernel associated with a corner contained in the interior of a polygon [21][14][22][23]. While the whole star region provides LOS for all points in a single corner environment, the star region in polygons must be carefully cropped to ensure complete coverage using polygon reduction [24]. In Figure 1, the region bounded by the dashed lines around c_4 is the star region associated with it. If a guard lies in the star region of a corner, the intruder cannot break the line-of-sight with the guard around the corner [21]. Therefore, if the star region associated with every corner in the polygon contains a guard, the entire polygon can be covered. Moreover, the number of guards required to cover the polygon can be reduced by deploying them in regions in which multiple star regions intersect.

A *star tile* is defined as an intersection of multiple star regions. We use $\langle c_1, \ldots, c_i \rangle$ to denote a star tile formed by intersections of star regions of corners $c_1, \ldots, c_i$. Corners $c_1, \ldots, c_i$ are called the *members* of the star tile. Figure 1 shows the star region associated with the corners in a polygon and the star tiles with their members. A star tile is said to be *covered* if it contains a guard. An intruder cannot break the LOS with a guard around a corner if the corner is a member of the star tile covered by the guard. Consider a collection of star tiles in a polygon. A collection of star tiles is called a *tile cover* if the union of members of the collection is the set of all corners in a polygon. A polygon can be covered by placing a guard in each tile which is a member of a tile cover. The cardinality of a tile cover is defined as the number of star tiles in the tile cover. Reducing the cardinality of a tile cover reduces the number of guards required to cover the polygon. A tile cover with the minimum cardinality is called a *minimal tile cover*. The next section presents an algorithm to efficiently generate a minimal tile cover for a polygonal environment.

IV. Computing Minimal Tile Cover

Computing all the star tiles in a polygon is expensive ($O(2^n)$) as it involves computing the intersection of all star regions. In this section, we present a modification of the *Apriori Algorithm* [25][26], used for mining frequent item sets [27][28][29] and association rules in relational databases, to efficiently compute the star tiles in a polygon. The algorithm prunes out associations which are not possible without having to check for actual associations in the databases.

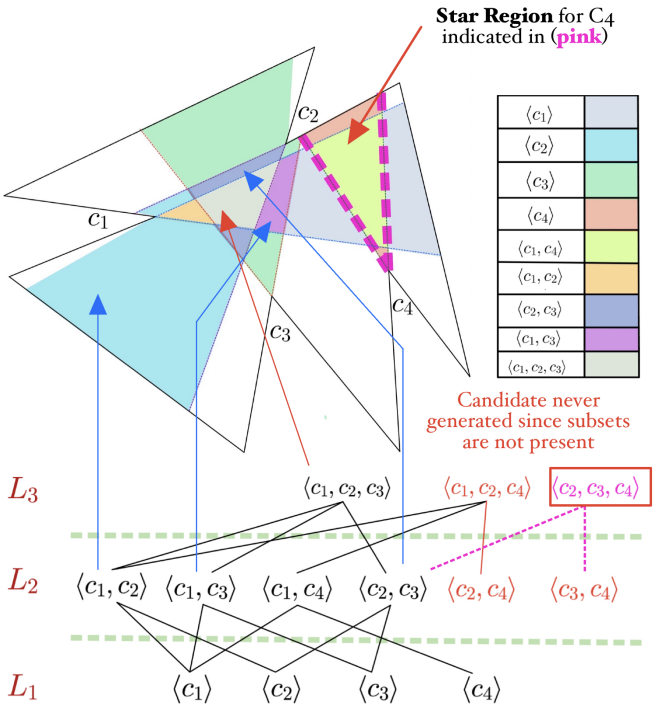


Fig. 1: Figure shows the generation of the Apriori structure for a polygonal environment

First, we define an itemset which are basic building blocks of the Apriori algorithm [8]. An item is an entity for an association. A collection of a set of items is called an itemset. If any itemset has k items it is called a k-itemset. Item sets play an important role in data-mining where frequent itemsets are recovered from huge datasets. In our case, each tile can be represented as an itemset where the member corner can be considered as items (we will be using the terms tile and star tile interchangeably for the rest of your paper). For example, a tile can be encoded as $\langle c_1, ..., c_j \rangle$, wherein the set of participating corners $c_1, ..., c_j$ are the individual items. A star region associated with a single corner c_i is represented as itemset $\langle c_i \rangle$.

Algorithm 1 Apriori Algorithm for region intersections

Input: T Set of corners
Output: L Apriori Structure
1: **function** APRIORI(T)
2: $L_1 \leftarrow \{\langle c\rangle | c \in T\}$
3: $k \leftarrow 2$
4: **while** $L_{k-1} \neq \emptyset$ **do**
5: $C_k \leftarrow$APRIORI GEN(L_{k-1}, k)
6: $L_k \leftarrow \{m \mid m \in C_k \wedge m \in P\}$
7: $k \leftarrow k+1$
8: **end while**
9: **return** UNION(L_k)
10: **end function**

Algorithm 2 Apriori Item set generator

Input: D All item-sets in L_{k-1}, k Current level
Output: J Set of candidate item-sets
1: **function** APRIORI GEN(D, k)
2: $J \leftarrow emptylist()$
3: **for** all $p \subseteq D, q \subseteq D$ where $p_1 = q_1, ..., p_{k-1} = q_{k-1}$ and $p_{k-1} < q_{k-1}$ **do**
4: $s \leftarrow p \cup \{q_{k-1}\}$
5: **if** $u \subseteq s, \forall u \in D$ **then**
6: J.ADD(s)
7: **end if**
8: **end for**
9: **return** J
10: **end function**

Figure 1 illustrates the star tile generation for the polygon with corners $\{c_1, c_2, c_3, c_4\}$. Algorithm 1 is initialized with L_1 that has all the 1-itemsets $\langle c_1\rangle, \langle c_2\rangle, \langle c_3\rangle, \langle c_4\rangle$. In L_1, all the itemsets correspond to the star regions of the corners since star regions are 1-item star tiles. Each layer L_k is a collection of itemsets of length k which correspond to star tiles in the actual environment formed by the intersection of k corners. However, enumerating all the itemsets in L_k that correspond to non-empty intersections of star regions from k corners is $O(\binom{n}{k}kn^2)$ which is very expensive[1]. To reduce the computational burden of generating L_k, the following steps are implemented in Algorithm 1 to eliminate redundant members:

1) First a candidate itemset is generated by performing a union operation on items in L_{k-1} which have $k-2$ common items between them (Algorithm 2, Line 4). For example, $\langle c_1\rangle$ and $\langle c_2\rangle$ are combined to obtain the candidate itemset $\langle c_1, c_2\rangle$. Similarly, for $\langle c_1, c_2, c_3\rangle$ in L_3, the candidate itemset is first generated by **union** of $\langle c_1, c_2\rangle$ and $\langle c_2, c_3\rangle$ since c_2 is present in both ($k-2$ common elements where $k=3$).

[1]Computing the intersection of two polygons has a worst-case time complexity of $O(n^2)$ where n is the number of vertices of the polygon. Moreover, the number of candidate itemsets at each level is $O(\binom{n}{k})$. Therefore, the overall computational complexity of building level L_k is $O(\binom{n}{k}kn^2)$

2) Next the validity of a candidate itemset is performed by checking the subset in L_{k-1}(Algorithm 2, Line 6). For a k-itemset to be a candidate member of L_k, all combinations of $k-1$ members of the k-itemset must be present at L_{k-1}. This step is called the main pruning step since we can prune non-existent intersections inside the polygons by examining the elements in L_{k-1}. For example, $\langle c_1, c_2, c_4\rangle$ is generated as a candidate set for L_3 in L_3. However, it does not show up in the apriori structure since $\langle c_2, c_4\rangle$ is absent from L_2.

For the polygon shown in Figure 1, Algorithm 1 halts at L_3 since it contains a single itemset. Even though the worse-case complexity of Algorithm 1 does not change, the screening steps give us a much better performance in real time (empirically shown in Section VII).

Algorithm 3 Algorithm for Minimal Tile Cover

Input: k Height of structure, L Apriori Structure
Output: U Minimal star tiles
1: **function** MINIMAL TILES(L, k)
2: $U \leftarrow \phi$
3: **while** $k \geq 0$ **do**
4: **for** $i \in L_k$ **do**
5: **if** $\forall x \in i, \nexists y \in U \wedge x \notin y$ **then**
6: Add i to F
7: **end if**
8: **end for**
9: Create a graph $G = (V, E)$ s.t. $V = \{v | \forall v \in F\}$ and $E = \{(u, v) | u, v \in F \wedge u \cap v \neq \phi\}$.
10: $j \leftarrow 0$
11: **while** *valid* nodes remain in G **do**
12: Add nodes with degree j to U and set linked nodes to *invalid*
13: $j \leftarrow j+1$
14: **end while**
15: $k \leftarrow k-1$
16: **end while**
17: **return** U
18: **end function**

Algorithm 3 generates a tile cover on an Apriori structure. Since higher levels of the Apriori structure combine more corners compared to lower levels, the algorithm iterates from the top to the bottom of the Apriori structure. This problem can be shown to be NP-Hard using the Independent Systems problem [30]. Hence we use a greedy strategy to find a tile cover.

Algorithm 3 can also be initialized at lower levels of the Apriori structure to reduce the time required to compute a tile cover. However, initiating Algorithm 3 at lower levels may not yield the optimal tile cover. **Figure 2 shows the trade-off between the computation time and cardinality of the tile cover** when Algorithm 3 is initialized at different levels of the Apriori structure of a polygon.

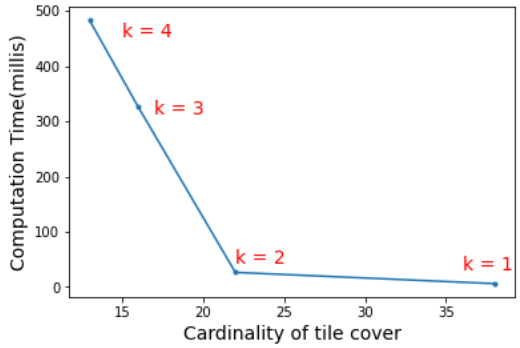


Fig. 2: Tradeoff between cardinality of the optimal tile cover and computation time for a polygon with 100 vertices using Algorithm 3. The figure shows the results for initialization of the algorithm at different levels (L_k) of the Apriori structure obtained from Algorithm 1.

V. Target tracking on Tile Graphs

In the previous section, we presented an algorithm to construct a Minimal Tile Cover (MTC) inside a polygon. If a single guard covers each star tile of MTC, the entire polygon is covered by the guards. However, if the number of guards is less than the cardinality of the MTC, the guards have to move between the star tiles to maintain LOS with the intruder. In this work, we propose tracking strategies for a guard which is constrained to move between at most two star tiles in the MTC.

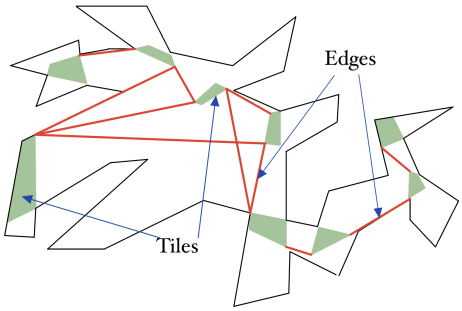


Fig. 3: Figure shows a tile cover and the corresponding tile graph for a polygonal environment.

Given a tile cover, we construct a *tile graph* that encodes the connectivity of the tiles based on mutual visibility. A tile graph $G = (V, E)$ is defined as a graph where each vertex $v \in V$ represents a star tile. An edge $e \in E$ exists between two vertices if a line-of-sight exists between any two points in the star tiles corresponding to the vertices[2]. Additionally, a self-loop is added at each vertex. For any polygon, the tile graph is connected. If a guard is deployed on a self-loop of the tile graph, it is placed inside the star tile corresponding to the vertex incident to the edge. If a guard is deployed on an edge connecting to distinct vertices in the tile graph, it moves on a line segment connecting the star tiles corresponding to the vertices incident to the edge.

The problem of covering the tile graph with minimum number of edge guards reduces to the minimum vertex cover problem which is known to be NP-Hard [35][36][37]. However, several approximation algorithms [38][39][40][41] exist for the minimum vertex cover problem. It has been shown that the approximation ratio for the vertex cover problem has a lower bound of $\sqrt{2}-\varepsilon$ for any $\varepsilon > 0$ [42]. A vertex cover with an approximation factor of $2 - \Theta\left(\frac{1}{\sqrt{\log|V|}}\right)$ can be obtained using the algorithm proposed in [43].

Next, we present the control strategy for an edge guard to track an intruder as it moves between two tiles where the corners for each tile lie in separate half-planes which is inspected using Support-Vectors [44]. Refer to Figure 4a which shows a coordinate system attached to one end of the edge connecting two tiles on which a guard is deployed. Let x_g denote the position of the guard along the x-axis. Let x_e and u_e denote the component of the position and the velocity of the evader along the edge (i.e., the x-axis of the coordinate system). The control law for the guard to track the evader is given by the hybrid automaton in Figure 4b.

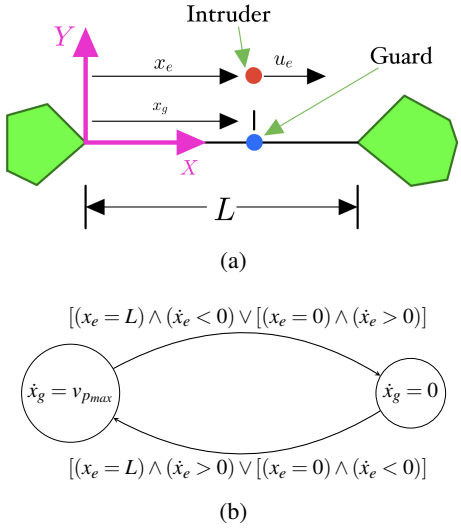


Fig. 4: (a) Figure shows the coordinate system between two tiles as a guard moves between them (b) Figure shows the automaton governing the movement of the guard with respect to the movement of the intruder.

Deploying a guard on an edge to successfully track the target involves the following steps:

1) Identify the points in the star tiles between which the guard travels on a straight line while tracking the intruder. This can be computed using the visibility graph.
2) A proportional control law for the guard that depends on the current position of the intruder relative to the two endpoints of the guard's path can be used to track the intruder. In order to track successfully, the speed of the guard should be $v_{p_{max}} = v_e \frac{L}{d_{min}}$ where L is the minimum distance between any two points in polygons T_1 and T_2

[2]For two tiles, if mutual visibility of the vertices is occluded due to the presence of a obstacle (for example, a narrow passage), an additional angular ray sweep [31] on the visibility graph [32][33][34] containing the vertices of the polygon and the tiles can establish visibility.

(as shown in Figure 4a), and d_{min} is the minimum distance of the intruder from the set of corners associated with T_1 and T_2 [14].

3) An activation strategy for the guards to persistently track the intruder

The control law proposed in this section requires the guards and the intruder to have the same maximum speed. In fact, it can shown that this speed bound is tight. Refer to Figure 5. It can be seen that m static guards (placed at the base of each fringe) are necessary and sufficient to cover the environment. However, $m-1$ mobile guards can track an intruder if and only if their speed is at least equal to the speed of the intruder. This can be proved using a combed environment similar to the Fortress problem [12] while reducing the width of the corridor forcing the guard to travel at least v_e speed.

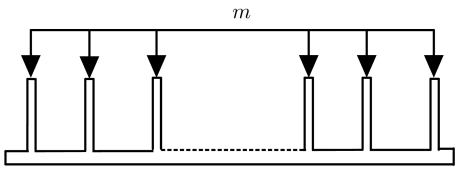


Fig. 5: A "comb" environment

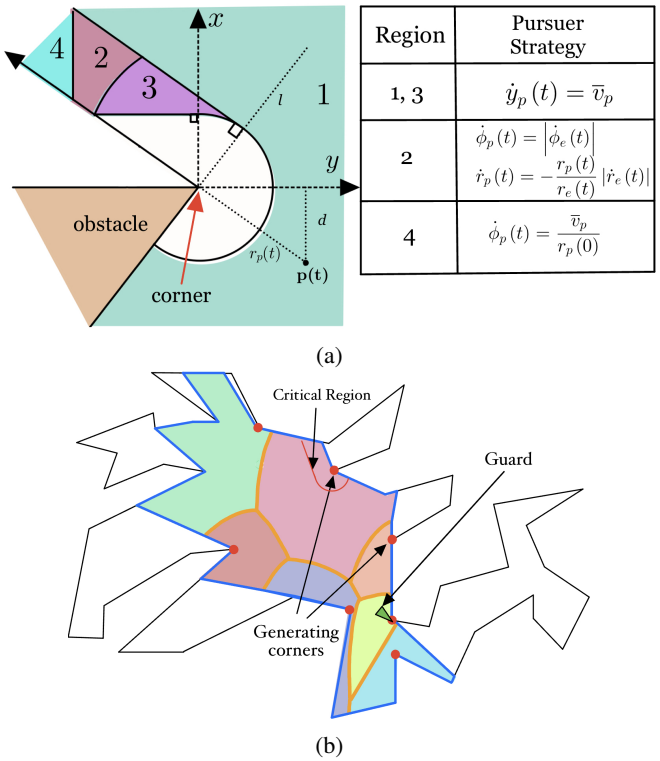
VI. Deployment Strategy for Persistent Tracking

In this section, we address the problem of tracking an intruder when a team of co-located mobile guards are deployed at an arbitrary point in the polygon. In [14], the authors addressed the problem of persistent tracking with a single mobile guard around a corner as a pursuit-evasion game. They provided a partition of the space (Figure 6a) around the corner based on the winning strategy for the guard (intruder) to track (break line-of-sight). A key result from the analysis in [14] is that if the intruder lies in Region 1 (shaded in green) of the partition, the optimal strategy of the guard is to minimize its distance from the star region. In this work, we extend this result to environment containing multiple corners.

Let p denote the location of the guard inside a polygon. Let $v_1,\ldots v_r$ denote the set of corners of $VP(p)$ for which p does not lie in their star region, where $VP(p)$ denotes the visibility polygon[3] at p. As stated in the previous paragraph, the pursuer induces a partition around each reflex vertex in the set $v_1,\ldots v_r$. Let l_i (generators of the voronoi partition) denote the boundary of Region 1 in the partition associated with vertex v_i. The *Tracking-based Voronoi Diagram* (TVD), a type of dynamic voronoi [45], is a partition of $VP(p)$ into cells induced by the generators $l_1,\ldots,l_r$. The cell (R_i) associated with l_i is given by the following:

$$R_i = \{x \in VP(p) | d(x,l_i) \leq d(x,l_j) \text{ for all } i \neq j\} \quad (1)$$

We propose a leader-follower relay-pursuit strategy [46] for the team of guards which is described as follows. Initially, all the guards in the team (including the leader) follow the direct line-of-sight to the intruder. One guard is chosen as the leader of the team. The choice may be random or it may depend on other high level mission specific goals.



Region	Pursuer Strategy
1, 3	$\dot{y}_p(t) = \overline{v}_p$
2	$\dot{\phi}_p(t) = \lvert\dot{\phi}_e(t)\rvert$ $\dot{r}_p(t) = -\frac{r_p(t)}{r_e(t)}\lvert\dot{r}_e(t)\rvert$
4	$\dot{\phi}_p(t) = \frac{\overline{v}_p}{r_p(0)}$

(a)

(b)

Fig. 6: (a) shows the partition around a corner based on the strategies of the winner of a pursuit-evasion game analyzed in [14]. (b) shows the cells of the TVD of a guard along with the generators in a polygonal environment.

The leader is responsible for maintaining a line-of-sight with the intruder at all times, and communicate the intruder's position to the rest of the team. As the leader pursues the intruder, it computes the TVD. If the intruder lies on the boundary of Region 1 of any corner in the leader's TVD (termed as a "critical event"), it follows the optimal policy around the corner (as given in Figure 6a) to ensure that the intruder does not break the line-of-sight around the corner. Once it reaches the boundary of the star region, it moves towards the star tile associated with corner. After a critical event occurs, the followers choose a new leader among themselves, and continue the tracking task.

The overall tracking strategy consists of two phases. In the first phase, the guards navigate from their initial position to a set of tiles that forms the minimal (or the $k-$minimal) tile cover using the strategy proposed in the previous paragraph. In the second phase, the guards track the intruder using the strategy proposed in the previous section.

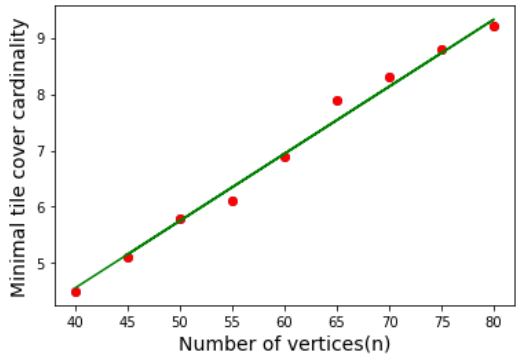
VII. Simulations and Empirical Analysis

In this section, we present empirical results on the performance of the algorithms proposed in our current work. An important parameter that dictates the efficacy of our technique is the number of guards that need to be deployed

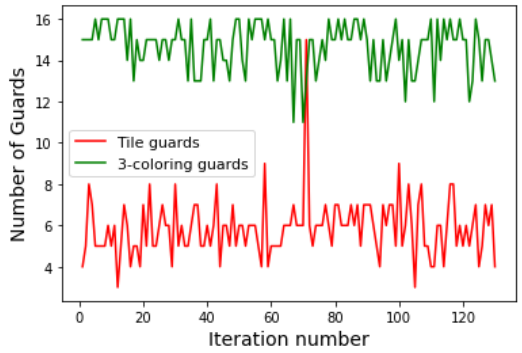
[3]The visibility polygon of p denotes the set of all points inside a polygon that are visible to p

to accomplish the tracking task which in turn depends on the number of tiles in the maximal tile cover. Figure 7a shows the result of implementing the Algorithm 3 on 9000 random polygons with n sides, where n varies in the interval $[40, 80]$. The plot shows that the correlation between the ratio of the number of tiles to the number of vertices is approximately **0.119** which is almost a third of $\frac{1}{3}$ (the worst case bound for static guard coverage).

Figure 7b compares the state-of-the-art 3-coloring solution using triangulation [12][47][48][49] and Algorithm 3 in terms of the number of guards deployed for 130 random polygons. This shows that our proposed technique leads to a fewer deployment of guards in comparison to covering the entire polygon with static guards for tracking applications.



(a)



(b)

Fig. 7: (a) Plot shows the correlation between the number of tiles using Algorithm 3, and the number of sides of the polygon for 9000 polygons chosen randomly (b) Plot shows the comparison between the number of tile guards for Algorithm 3 and the number of vertex guards needed to cover a polygon using the triangulation method.

Figure 8 shows the variation in the computation time required to obtain the optimal tile cover as the size of the polygon using two different techniques (i) Algorithm 3 (ii) a brute-force technique that computes all the intersections of star regions in the polygon. From the plots, we can clearly see that the Algorithm 3 is orders of magnitude faster than the brute-force technique. For a polygon with 150 vertices having 53 corners, Algorithm 3 takes **18.29 secs** to compute the optimal tile cover as opposed to the **92 years** it would take the brute-force method to compute the same.

Finally, Figure 9 shows a snapshot of the proposed technique being implemented on a team of drones in our testbed. Please refer to the video attachment for details regarding the implementation scenario.

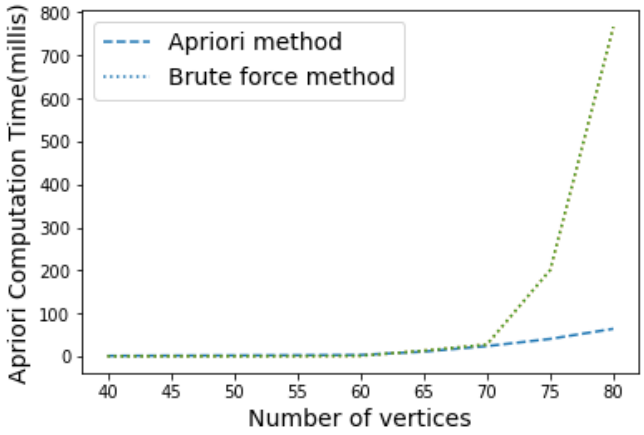


Fig. 8: Figure shows the comparison between the time taken to compute the tiles using our proposed Algorithm 3 and a brute force technique that computes the intersection between all possible star regions in the polygon.

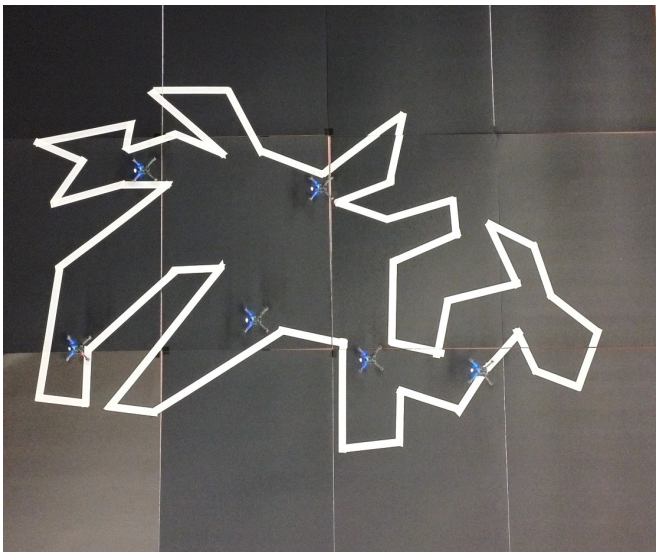


Fig. 9: Figure shows the testbed for implementing the proposed tracking technique on a team of crazyflie drones. Please refer to the video attachment for the complete simulation and implementation.

VIII. Conclusion and Future Works

In this work, we built a bridge between data mining and visibility-based tracking using a novel tiling scheme of the polygon. In the future, we plan to extend this technique to track multiple intruders in the environment. Dealing with motion and sensing constraints for the guards is another direction of future work. Extension of the technique for tracking in 3-d environments [50][51][52] is a promising direction for future research.

REFERENCES

[1] N. Zhang, W. Chunxue, Y. Wu, and N. Xiong, "An improved target tracking algorithm and its application in intelligent video surveillance system," *Multimedia Tools and Applications*, vol. 79, 06 2020.

[2] W. Du, T. Guo, J. Chen, B. Li, G. Zhu, and X. Cao, "Cooperative pursuit of unauthorized UAVs in urban airspace via multi-agent reinforcement learning," *Transportation Research Part C: Emerging Technologies*, vol. 128, p. 103122, 07 2021.

[3] J. Sandino, F. Vanegas Alvarez, L. Gonzalez, and F. Maire, "Autonomous UAVv navigation for active perception of targets in uncertain and cluttered environments," 03 2020.

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

[6] 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.

[7] W. pang Chin and S. C. Ntafos, "Optimum watchman routes," in *SCG '86*, 1986.

[8] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, "Privacy preserving mining of association rules," *Information systems (Oxford)*, vol. 29, no. 4, pp. 343–364, 2004.

[9] G. Piatetsky-Shapiro, "Discovery, analysis, and presentation of strong rules," in *Knowledge Discovery in Databases*, 1991.

[10] E. Bormashenko, I. Legchenkova, M. Frenkel, N. Shvalb, and S. Shoval, "Informational measure of symmetry vs. voronoi entropy and continuous measure of entropy of the penrose tiling. part ii of the "voronoi entropy vs. continuous measure of symmetry of the penrose tiling"," *Symmetry*, vol. 13, p. 2146, 2021.

[11] D. Banerjee and R. Inkulu, "Vertex guarding for dynamic orthogonal art galleries," *arXiv.org*, 2021.

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

[13] L. Palios, "Decomposition problems in computational geometry," 1992.

[14] 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

[15] G. Voronoi, "Nouvelles applications des paramètres continus à la théorie des formes quadratiques. premier mémoire. sur quelques propriétés des formes quadratiques positives parfaites." *Journal für die reine und angewandte Mathematik (Crelles Journal)*, vol. 1908, pp. 97 – 102.

[16] F. Aurenhammer, "Voronoi diagrams—a survey of a fundamental geometric data structure," *ACM Comput. Surv.*, vol. 23, pp. 345–405, 1991.

[17] 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

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

[19] 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

[20] 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.

[21] 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.

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

[23] T. Sorgente, S. Biasotti, and M. Spagnuolo, "Polyhedron kernel computation using a geometric approach," 2022. [Online]. Available: https://arxiv.org/abs/2202.06625

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

[25] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '93. New York, NY, USA: Association for Computing Machinery, 1993, p. 207–216. [Online]. Available: https://doi.org/10.1145/170035.170072

[26] *Frequent Pattern Mining edited by Charu C. Aggarwal, Jiawei Han.*, 1st ed., 2014.

[27] Y. Djenouri, A. Belhadi, and P. Fournier-Viger, "Extracting useful knowledge from event logs: A frequent itemset mining approach," *Knowledge-based systems*, vol. 139, pp. 132–148, 2018.

[28] Y. Xun, X. Cui, J. Zhang, and Q. Yin, "Incremental frequent itemsets mining based on frequent pattern tree and multi-scale," *Expert Systems with Applications*, vol. 163, p. 113805, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417420306217

[29] L. Palios, "Decomposition problems in computational geometry," 1992.

[30] S. Zhou, "Minimum partition of an independence system into independent sets," *Discrete Optimization*, vol. 6, no. 1, pp. 125–133, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1572528608000698

[31] E. Welzl, "Constructing the visibility graph for n-line segments in o(n2) time," *Information Processing Letters*, vol. 20, no. 4, pp. 167–171, 1985. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0020019085900444

[32] J. O'Rourke and I. Streinu, "Vertex-edge pseudo-visibility graphs: Characterization and recognition." 08 1997, pp. 119–128.

[33] H. Niu, A. Savvaris, A. Tsourdos, and Z. Ji, "Voronoi-visibility roadmap-based path planning algorithm for unmanned surface vehicles," *Journal of Navigation*, vol. 72, no. 4, p. 850–874, 2019.

[34] M. Overmars, E. Welzl, and M. T, "New methods for computing visibility graphs," *Proceedings of the Fourth Annual Symposium on Computational Geometry*, 08 2001.

[35] R. M. Karp, *Reducibility among Combinatorial Problems*. Boston, MA: Springer US, 1972, pp. 85–103. [Online]. Available: https://doi.org/10.1007/978-1-4684-2001-2_9

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

[37] J. Chen, I. A. Kanj, and G. Xia, "Improved upper bounds for vertex cover," *Theoretical Computer Science*, vol. 411, no. 40, pp. 3736–3756, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304397510003609

[38] K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, 01 1982, vol. 32.

[39] I. Dinur and S. Safra, "On the hardness of approximating minimum vertex cover," *Annals of Mathematics*, vol. 162, no. 1, pp. 439–485, 2005. [Online]. Available: http://www.jstor.org/stable/3597377

[40] A. Zhang, Y. Chen, Z.-Z. Chen, and G. Lin, "Improved approximation algorithms for path vertex covers in regular graphs." [Online]. Available: https://arxiv.org/abs/1811.01162

[41] B. Bresar, F. Kardos, J. Katrenic, and G. Semanisin, "Minimum k - path vertex cover," *Discrete Applied Mathematics*, vol. 159, no. 12, pp. 1189–1195, 2011.

[42] K. Subhash, D. Minzer, and M. Safra, "Pseudorandom sets in grassmann graph have near-perfect expansion," 10 2018, pp. 592–601.

[43] G. Karakostas, "A better approximation ratio for the vertex cover problem," *Electronic Colloquium on Computational Complexity (ECCC)*, vol. 5, 01 2004.

[44] 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

[45] E. Bakolas and P. Tsiotras, "The zermelo–voronoi diagram: A dynamic partition problem," *Automatica (Oxford)*, vol. 46, no. 12, pp. 2059–2067, 2010.

[46] V. R. Makkapati, W. Sun, and P. Tsiotras, "Optimal evading strategies for two-pursuer/one-evader problems," *Journal of guidance, control, and dynamics*, vol. 41, no. 4, pp. 851–862, 2018.

[47] D. Avis and G. Toussaint, "An efficient algorithm for decomposing a polygon into star-shaped polygons," *Pattern Recognition*, vol. 13, no. 6, pp. 395–398, 1981. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0031320381900029

[48] A. Kooshesh and B. Moret, "Three-coloring the vertices of a triangulated simple polygon," *Pattern Recognition*, vol. 25, p. 443, 04 1992.

[49] S. Ashur, O. Filtser, M. Katz, and R. Saban, *Terrain-Like Graphs: PTASs for Guarding Weakly-Visible Polygons and Terrains*, 01 2020, pp. 1–17.
[50] E. Lipka, "A note on minimal art galleries," 09 2019.
[51] S. Vaddi, D. Kim, C. Kumar, S. Shad, and A. Jannesari, "Efficient Object Detection Model for Real-time UAV Application," *Computer and Information Science*, vol. 14, no. 1, pp. 1–45, February 2021. [Online]. Available: https://ideas.repec.org/a/ibn/cisjnl/v14y2021i1p45.html
[52] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 21–37. [Online]. Available: https://doi.org/10.1007/978-3-319-46448-0_2