# Monitoring Access to User Defined Areas with Multi-Agent Team in Urban Environments

Manas Gupta, Ming C Lin, Dinesh Manocha, Huan Xu, and Michael Otte

***Abstract*— We present an algorithm that determines where the members of a multi-agent team or swarm should be deployed in order to efficiently monitor access to a user specified region of interest. Our algorithm attempts to minimize the number of agents required to guarantee that any incursion into the region of interest is detected. The algorithm works by analyzing the geometric structure of the environment, and placing agents at advantageous positions in the environment, such as bottlenecks, to create a defensive perimeter of agents alongside physical obstacles (e.g. buildings). We demonstrate the usefulness of the algorithm through experimental simulations in an urban environment, and show how the min-cuts subroutine (used to reduce the number of agents required) can be implemented in a distributed way across the multi-agent team to enable better solutions to be found more quickly.**

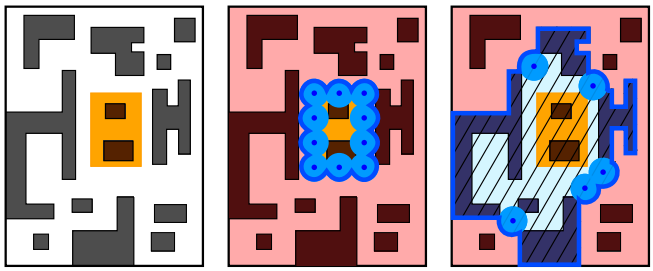
## I. Introduction

Autonomous robots are used for a variety of surveillance missions in both civilian and non-civilian contexts. From performing night-watch security in factories and shopping malls, to providing intelligence about humanitarian disasters and conflict zones; autonomous teams are often a cost effective way to enhance and extend human capabilities and/or to keep humans out of harm's way. We consider a non-standard twist on the multi-agent autonomous surveillance problem. While standard autonomous surveillance problems typically involve monitoring an entire area or finding/tracking all adversaries within it, **we are interested in autonomously monitoring *all routes of access into* an area**.

By simultaneously monitoring all access routes into an area (Figure 1), potential intruders can be detected before they enter the area. This is useful because it enables the multi-agent team to either prevent the incursion or, alternatively, to monitor the adversary's passage through the area. We formalize this problem as the *Isolation Region Surveillance Problem*, and study a variant in which the goal is to solve the problem with minimum number of agents. We call latter problem variant the *Minimum Robot Isolation Region Surveillance Problem*.

While related isolation region problems have been studied in a graph theoretic context [1], [2], [3], we are interested in extending such ideas to a three-dimensional geometric scenario. In particular, robots with sensors operate in a three-dimensional environment that contains polytopal obstacles, adversaries are restricted to ground-level movement, and the ground-level obstacles experienced by adversaries are equivalent to the polygonal footprints obtained by projecting obstacles down onto the ground-plane. Other assumptions include: Agents have reliable communication; adversaries have nonzero volume, the environment is adversary free until agents reach their assigned monitoring locations; and agents have downward-facing cameras (as with drones) that must be used below a maximum height.

*The authors are with the University of Maryland College Park. This work was supported by DARPA cooperative agreement HR00111820028 as part of DARPA OFFSET.



**A**: Area of Interest **B**: Naive Placement **C**: Our Method

Fig. 1. **A**: Map of an urban environment projected onto the ground-plane $\mathcal{X} \subset \mathbb{R}^2$ with obstacles $\mathcal{X}_{obs}$ (dark) and a user defined region of interest $\mathcal{X}_{int}$ (orange) for which access must be monitored or guarded. **B**: Many agents (dark dots) may be required to cover the boundary of $\mathcal{X}_{int}$ with sensors (blue discs). **C**: Our method uses fewer agents by placing agents at the bottlenecks created by obstacles. This creates a safe region of isolation $\mathcal{X}_{iso}$ (shaded blue area) containing the region of interest, $\mathcal{X}_{int} \subset \mathcal{X}_{iso}$.

One difference between our work and previous work is that a *geometric* model of the (urban) environment is provided as input. This model is discretized as a function of the camera height and field-of-view to obtain both: (1) A triangulation of the ground-surface, and (2) a graph that encodes the connectivity between the triangles. A min-cut of (2) provides a set of triangles that, combined with obstacles, separates the surveillance area from the environmental boundary. We prove that placing agents to observe each triangle in this set guarantees that any adversary moving from the environmental boundary to the surveillance area will be observed.

This paper is organized as follows: Section II contains a discussion of related work, nomenclature is described in Section III, and a formal problem definition appears in Section IV. The algorithm and it analysis appear in Section V and VI, respectively. Experiments involving a simulated urban environment appears in Section VII, a discussion of results in Section VIII, and our conclusions in Section IX.

## II. Related Work

Previous methods that consider graph topology when placing agents for surveillance or capture missions include [4], [5], [6] and [7]. In [4] the search domain is represented as a traversability graph, and the graph is modified into a tree by placing robots on edges to remove cycles. In [5], a

surveillance algorithm is proposed that solves a contiguous search problem and calculates the minimum number of agents required to capture an intruder. Similarly, [6] uses a decentralized control strategy to patrol user specified points, while agents coordinate in a decentralized way. In [7], a distributed multi-agent system is tasked with monitoring a graph-based environment, some of which may be unknown at the beginning of the mission. Our work differs from [4], [5], [6], [7] in three ways: we consider the three-dimensional geometry of the environment, we seek to isolate a region of interest from the map's boundary, and the connectivity graph is created at run-time as a function of the environment's geometry and the robots' sensor properties.

An "area of interest" is used in [8], which minimizes the time required to actively cover the entire area by computing the minimal convex cover and then finding shortest multi-path using Tabu Search. Work in [9] solves a related "graph-clear" problem that calculates a multi-path such that any intruders within the area will be detected. Our work differs from [8] and [9] in that we are interested in detecting adversaries before they enter the region of interest.

Other work on autonomous surveillance considering camera sensor properties includes: [10], [11], [12]. Bora *et al* [10] demonstrate surveillance over a roadmap by detecting the intruders in the environment using a network of autonomous vehicles with on board cameras. Semsch *et al* [11] generate trajectories for each UAV by considering the on-board camera's field of view, and then minimizing the average time between consecutive views of points in the area. Geng *et al* [12] solve a similar problem; a set of vantage points are calculated (placing a camera at each point would enable the entire area to be viewed at once), and individual paths between vantage points are calculated based on the assignment of vantage points to UAVs. While [10], [11] and [12] are interested in moving through the environment to continuously monitor an entire area, we are interested in guaranteeing that all adversaries are observed before they enter a particular area.

Surveillance is closely related to coverage and target search, which may involve either actively moving [13], [14], [15] or with stationary [16], [17] agents. In [17] Kazakis *et al* demonstrate an approach to guard a 2D area with small number of mobile agents having on board camera with limited visibility. The survey paper [18] contains pointers to additional related work.

Our work uses a Delaunay triangulation and its Vornoi dual. Delaunay triangulation has previously been used to solve various robotics problems, as in [19] and [20] for computing the collision free trajectories. In [21] Delaunay triangulation is used to model discrete point data from the segmented image for information extraction of cultivated land by UAVs. Other applications can be found in [22], [23]. Delaunay triangulation is the dual of the Voronoi graph, a relationship that has been central to applications of path planning, e.g., [19], [20], [24]. A Delaunay triangulation of a point set can be modified if points are inserted or removed from the set [25], [20].

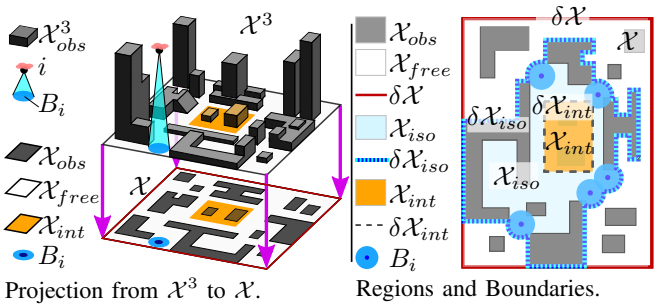



Fig. 2. Various quantities related to spatial regions in $\mathcal{X}^3$ and $\mathcal{X}$ (left) and $\mathcal{X}$ (right).

## III. Nomenclature

We now define the nomenclature used throughout this paper (Figure 2 depicts selected quantities). The three-dimensional *workspace* in which the robots operate is denoted by $\mathcal{X}^3 \subset \mathbb{R}^3$. The *obstacle space* $\mathcal{X}^3_{obs}$ and *free space* $\mathcal{X}^3_{free}$ are the subsets of $\mathcal{X}^3$ that contain obstacles and do not contain obstacles, respectively; $\mathcal{X}^3_{obs}, \mathcal{X}^3_{free} \subset \mathcal{X}^3$. The projections of $\mathcal{X}^3$, $\mathcal{X}^3_{obs}$, and $\mathcal{X}^3_{free}$ straight down onto the (two-dimensional) ground-plane are denoted $\mathcal{X}$, $\mathcal{X}_{obs}$, and $\mathcal{X}_{free}$, respectively. We refer to projections from $\mathcal{X}^3$ onto $\mathcal{X}$ as *footprints*. Assuming that each robot $i$ has a downward facing camera with a conic field of view, the footprint of this field of view is a *sensor disc* $B_i \subset \mathcal{X}$. While we assume $B_i$ is a disc, our method can also be used with other sensor footprint shapes by considering the largest disc that they contain. We assume a robot sensor footprint is larger than the projection of the robot itself, i.e., its volume, onto the ground-plane.

The area that the user decides to monitor is called the *Region of Interest* and is denoted $\mathcal{X}_{int} \subset \mathcal{X}$. The perimeter of $\mathcal{X}_{int}$ is called the *Region of Interest Boundary*[1] and denoted $\delta\mathcal{X}_{int}$. Note that $\delta\mathcal{X}_{int}$ separates $\mathcal{X}_{int}$ from the rest of $\mathcal{X}$ and is guaranteed by Jordan curve theorem. The *map boundary* is the one-dimensional perimeter of the two-dimensional ground projection $\mathcal{X}$ and denoted $\delta\mathcal{X}$.

An important region calculated by our method is the *Region of Isolation* $\mathcal{X}_{iso}$, where $\mathcal{X}_{int} \subset \mathcal{X}_{iso} \subset \mathcal{X}$. The *Region of Isolation Boundary* is denoted $\delta\mathcal{X}_{iso}$ and is—by definition—created from (and covered by) a combination of obstacles and agent sensor discs. Formally, $\delta\mathcal{X}_{iso} \subset \mathcal{X}_{obs} \cup \left(\bigcup_{i\in[1,n]} B_i\right)$. Indeed, the main purpose of our method is to place $B_i$ to create $\delta\mathcal{X}_{iso}$ with interior $\mathcal{X}_{iso}$ such that $\mathcal{X}_{int} \subset \mathcal{X}_{iso} \subset \mathcal{X}$—and to do so using the fewest number of agents possible.

We assume that adversaries travel at ground-level and are constrained to travel through $\mathcal{X}_{free}$. A valid adversary *path* $\pi$ is a curve through $\mathcal{X}_{free}$ that moves from a start position on the boundary of the map, $x_{start} \in \delta\mathcal{X}$, to a goal

[1]In practice, $\delta\mathcal{X}_{int}$ can be defined by a cycle of line segments or other well behaved point-wise continuous and closed curves.

position within the area of interest, $x_{goal} \in \mathcal{X}_{int}$. Formally, $\pi$ is a continuous mapping $\pi : [0,1] \rightarrow \mathcal{X}_{free}$ such that $\pi(0) = x_{start} \in \delta\mathcal{X}$ and $\pi(1) = x_{goal} \in \mathcal{X}_{int}$.

Our method relies on a subroutine that computes a *Delaunay Triangulation* $D$ with respect to a set of discrete points $P \in \mathcal{X}$, where $P$ is either a random set calculated iteratively, or chosen from a predefined lattice. Triangle regions within the Delaunay Triangulation are denoted $T \in D$. The dual[2] of a Delaunay Triangulation is a Voronoi Partitioning, and the latter provides an undirected graph that encodes the connectivity between the triangle regions in the Delaunay Triangulation. In general, a graph $G = (V, E)$ consists of a set of *vertices* $v \in V$ and edges $e \in E$. Two vertices $v$ and $u$ are considered *neighbors* if there exists an edge $e = (u, v)$ that connects them. In an *undirected* graph, $e = (u, v) \in E \iff e = (v, u) \in E$. There is a one-to-one correspondence between triangles in the Delaunay Triangulation $T_j \in D$ and nodes in the Voronoi graph $v_j \in V$.

*Karger's Min-Cut Theorem* (from [1]): A Min-cut of an undirected graph $G = (V, E)$ is a partition of two vertices $v, u$ into two non empty group of vertices $V_1$ and $V_2$ such that $V = V_1 \cup V_2$ and $V_1 \cap V_2 = \emptyset$ with minimum number of edges crossing between them.

## IV. Problem Definition

In this section we formally define the two problems that we investigate. Recall that the user defines the region of interest $\mathcal{X}_{int}$. By definition, $\delta\mathcal{X}_{iso}$ has the property that $\delta\mathcal{X}_{iso} \subset \mathcal{X}_{obs} \cup \left(\bigcup_{i\in[1,n]} B_i\right)$.

**Problem 1,** The *isolation region surveillance problem*: Can $n$ robots be placed to form $\delta\mathcal{X}_{iso}$ such that $\mathcal{X}_{int} \subset \mathcal{X}_{iso} \subset \mathcal{X}$? which is a trivial condition to monitor access to $\mathcal{X}_{int}$. If a valid $\mathcal{X}_{iso}$ is found, then by construction, any valid adversary path (from the map boundary into the region of interest) necessarily intersects $\delta\mathcal{X}_{iso} \cap \mathcal{X}_{free}$, and by construction for all $x \in \delta\mathcal{X}_{iso}$ such that $x \in \delta\mathcal{X}_{iso} \cap \mathcal{X}_{free}$ it is guaranteed that $x \in B_i$ for some $i$.

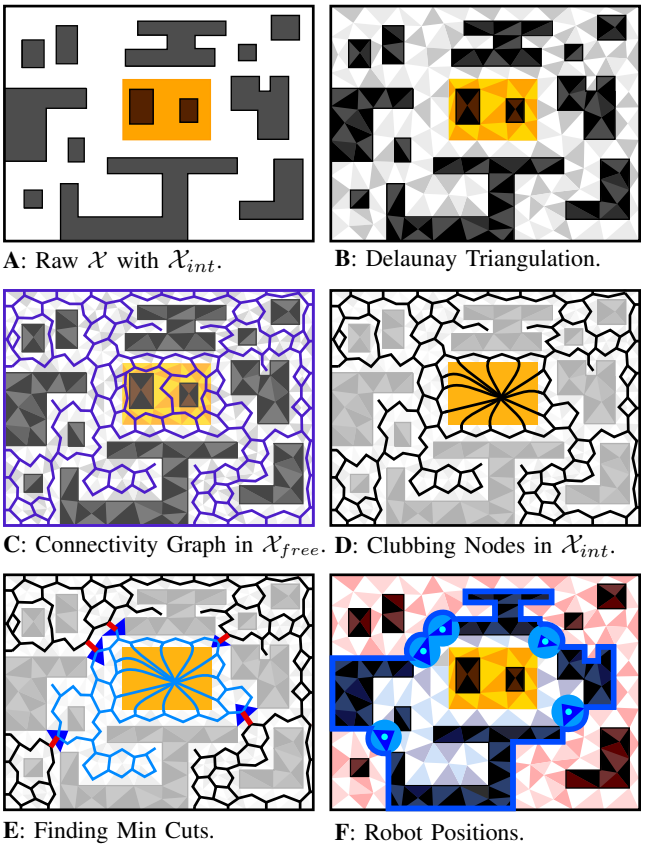
**Problem 2,** The *minimum robot isolation region surveillance problem*: Find the minimum $n$ that solves Problem 1.

## V. Algorithm

Pseudo code for our method appears in Algorithm 1, and the geometric and graph theoretic effects of the major subroutines are illustrated in Figure 3.

The algorithm starts by using $\texttt{Footprint}(\mathcal{X}^3)$ to extract the ground-plane free space $\mathcal{X}_{free}$ and the ground-plane obstacle space $\mathcal{X}_{obs}$ from the geometry of the work space $\mathcal{X}^3$ based on buildings in the environment, etc. (line 1). An initial point set $P$ is sampled on the interface of $\mathcal{X}_{free}$ and $\mathcal{X}_{obs}$, as well as along the boundaries of the region of interest and map, $\delta\mathcal{X}_{int}$ and $\delta\mathcal{X}$, respectively. The initial sampling is accomplished using $\texttt{InitPoints}(\mathcal{X}_{free}, \mathcal{X}_{obs}, \delta\mathcal{X}_{int}, \delta\mathcal{X})$, line 2. Along each boundary, consecutive points must be

[2]Each can be calculated from the other.



**A**: Raw $\mathcal{X}$ with $\mathcal{X}_{int}$. **B**: Delaunay Triangulation. **C**: Connectivity Graph in $\mathcal{X}_{free}$. **D**: Clubbing Nodes in $\mathcal{X}_{int}$. **E**: Finding Min Cuts. **F**: Robot Positions.

Fig. 3. The major steps in our method are depicted in **A**-**F**. Given a raw map of the environment and a user defined region of interest $\mathcal{X}_{int}$ (orange) within the map, we find the Delaunay triangulation (**B**) that is based on robot sensor radius, and the resulting connectivity graph (**C**, purple) which can found from the subset of the triangulation's Voronoi graph dual (in particular, the subset of the dual graph in $\mathcal{X}_{free}$). All nodes in the region of interest $\mathcal{X}_{int}$ are clubbed into a single *source* node (**D**). All nodes along the map boundary are also clubbed into a single *sink* node (not shown). Given the graph, source, and sink, we find the min-cuts—a set of edges (**E**, red) who's removal will separate the source and sink in two different connected components of the graph (**E**, light blue and black, respectively). The min-cuts are associated with triangles (**C**-**D**, dark blue), and each triangle is guaranteed by construction to fit inside a robot's sensor disc (**E**, dark blue) and to be un-occluded by obstacles. Placing robots accordingly, sensor discs and obstacles define the boundary of the region of isolation $\mathcal{X}_{iso}$ (**E**, blue) that contains the region of interest, $\mathcal{X}_{int} \subset \mathcal{X}_{iso}$.

**Algorithm 1:** $\texttt{monitorAccess}(\mathcal{X}^3, \mathcal{X}_{int})$

1 $(\mathcal{X}_{free}, \mathcal{X}_{obs}) \leftarrow \texttt{Footprint}(\mathcal{X}^3)$ ;
2 $P \leftarrow \texttt{InitPoints}(\mathcal{X}_{free}, \mathcal{X}_{obs}, \delta\mathcal{X}_{int}, \delta\mathcal{X})$ ;
3 **while exists** $T \in D$ s.t. $\texttt{circle}(T) > B$ **do**
4 $\quad P \leftarrow P \cup \{\texttt{SamplePoint}(\mathcal{X}_{free}, \mathcal{X}_{obs}, T)\}$ ;
5 $\quad D \leftarrow \texttt{IterativeDelaunay}(P)$ ;
6 **end**
7 $G \leftarrow \texttt{VornoiDual}(D)$ // $(V, E) = G$ ;
8 $(G, v_S, v_T) \leftarrow \texttt{MergeNodes}(G, \mathcal{X}_{int}, \delta\mathcal{X})$ ;
9 AgentPositions $\leftarrow \texttt{MinCuts}(G, v_S, v_T)$ ;
10 **return** AgentPositions ;

sampled no further from each other than the width of a robot sensor disc. Next, $P$ is iteratively expanded until all triangles $T$ in its Delaunay Triangulation $D$ have circumcircles[3] $\texttt{circle}(T)$ that are no larger than a robot sensor disc $B$ (lines 3-4). In Section VI we show that enforcing the condition that circumcircle centers are located within their corresponding triangles is a sufficient (but not necessary) condition to guarantee correctness and resolution completeness. While this can be done in a variety of ways, we recommend sampling new points based on the triangle that has the largest circumcircle (though care must be taken to add more boundary points when center of the circumcircle is outside the corresponding triangle).

Once an appropriate Delaunay Triangulation has been computed, we calculate the connectivity graph $G_v$ between the triangles using its the Voronoi dual (line 7). Because the graph is the dual of Delaunay triangulation, each triangle is associated with exactly one circumcircle, and so this provides the connectivity structure that we require.

The graph is post-processed by merging all nodes along the map boundary into a single source node $v_S$, and merging all nodes in the region of interest into a single sink node $v_T$; the subroutine $\texttt{MergeNodes}(G, \mathcal{X}_{int})$ is responsible for performing both of these merging (line 8).

The final agent positions are calculated using $\texttt{MinCuts}(G, v_S, v_T)$ and then returned (lines 9-10). A variety of algorithms exist that can perform the s-t (source-sink) variant of min-cuts that we require. In our experiments we use Karger's Min-cut algorithm because it can be implemented in a distributed fashion across the robot team. Karger's Min-cut algorithm assumes that edges are assigned weights, and each edge is given an equal weight of 1.

Final agent locations represent the circumcircle centers of the corresponding triangles in the Delaunay Triangulation, such that an agent deployed at a particular circumcircle center have the entire triangle within its sensor disc. Moreover, the union of obstacles and all triangles containing agents creates a boundary around the region of interest (Figure 3-F). Formally, we are guaranteed the existence of a $\delta\mathcal{X}_{iso}$ such that $\delta\mathcal{X}_{iso} \subset \mathcal{X}_{obs} \cup \left(\bigcup_{i\in[1,n]} T_i\right)$, and by construction $T_i \subset B_i$ for all $i \in [1, n]$.

## VI. Analysis of Algorithmic Properties

The algorithm in Section V efficiently solves Problem 1, and finds an approximate solution for Problem 2. The solution of problem 2 is an approximation for two reasons. First, the discretization of the map into triangles may not be optimal for the particular environment considered. Second, Karger's min-cut algorithm is stochastic in nature, and (if used) provides an approximate solution that is also stochastic in nature. We now discuss the algorithmic properties of the algorithm in more detail, considering both what happens when Karger's min-cut algorithm is used, and also (separately) the case when an optimal min-cuts algorithm is used.

[3]The circumcircle $\texttt{circle}(T)$ of a triangle region $T$ is the unique circle defined such that the three vertices of the triangle are located on the circle.

Assuming that balls and simplicies are closed sets, the following proposition is a well known fact from geometry.

*Proposition 1:* In $d$-dimensional space, the volume of any simplex is a subset of the volume of the ball that is uniquely defined by assuming that the $(d+1)$ vertices of the simplex are located on the surface of the ball.

In $\mathbb{R}^2$ a simplex is a triangle $T$, and a ball is a disc $B$. Thus, Proposition 1 implies that $T \subset B \subset \mathbb{R}^2$. By definition, the boundary of the particular disc $B$ that is uniquely defined by a triangle $T$ is called the triangle's circumcircle.

The following (Proposition 2) is true by the construction of our algorithm.

*Proposition 2:* For each $T \in D$, the circumcircle of $T$ is smaller than a robot's sensor disc, $T \subset B_i$ for all $i$.

The following (Lemma 1) is a building block that we will use shortly.

*Lemma 1:* Given an absence of local obstacle occlusions between $T_i \in D$ and the center of the circumcircle of $T_i$, a robot $i$ located at the center of the circumcircle of $T_i$ will have $T_i \subset B_i$.

*Proof:* This follows directly from Propositions 1 and 2. □

*Lemma 2:* If the center of the circumcircle of $T_i$ is located within $T_i \subset D \cap \mathcal{X}_{free}$, then a robot $i$ placed at the center of the circumcircle of $T_i$ will have a full view of $T_i$.

*Proof:* The definition of convexity[4], the fact that triangles are convex, and our assumptions on the projection from $\mathcal{X}^3$ to $\mathcal{X}$ means that there are no obstacles from $\mathcal{X}^3_{obs}$ occluding subsets of $T_i$ from $i$, and so Lemma 1 may be applied. □

On the other hand, we also note the following:

*Lemma 3:* If the center of the circumcircle of $T$ is not located within $T$, and $T \subset \mathcal{X}_{free}$, then it is possible that the view of $T$ may be occluded by $\mathcal{X}^3_{obs}$.

*Proof:* This happens, e.g., whenever the circumcircle center of a triangle and the triangle are separated by an obstacle. □

The following is a Corollary of Lemma 2.

*Corollary 1:* If, for all $T \in D$ the center of the curcumcircle of $T$ is located within $T$, then any adversary movement between two adjacent triangles $T_1, T_2 \in D \cap \mathcal{X}_{free}$ will be detected by an agent that is placed at the center of either of the corresponding circumcircles of $T_1$ or $T_2$.

*Remark:* A minimum $s-t$ cut results in the source $s$ and target $t$ nodes being in two separate sub graphs. This is required for our method. There are many versions of the global min-cut algorithms which run in polynomial time, e.g., Stoer and Wagner [3].

*Theorem 1:* If, for all $T \in D$ the center of the circumcircle of $T$ is located within $T$, then our method will place agents such that an adversary moving from the map boundary to the region of interest will be observed.

*Proof:* The dual between the Delaunay Triangulation and Voronoi graph (which is homomorphic to our un-clubbed connectivity graph), combined with our assumption of nonzero adversary volume and the fact that clubbing nodes

[4]A convex shape is defined by the property that for any two points $x_1$ and $x_2$ in the shape, the line segment $\overline{x_1 x_2}$ is fully contained in the shape.

in $\mathcal{X}_{int}$ changes the graph only within $\mathcal{X}_{int}$ (and clubbing map boundary nodes is similarly benign), guarantees that a min-cuts solution (indeed, even a sub optimal approximation to the min-cut) will return a set of edges, such that removing the edges will disconnect the region of interest $\mathcal{X}_{int}$ from the map boundary $\delta\mathcal{X}$. Placing agents at either end of each edge in the removal set—i.e., at the center of the circumcircle of either of the two triangles associated with the edge—guarantees that an adversary moving from the map boundary to the region of interest will be observed. □

We now consider correctness and resolution completeness, assuming that "*resolution*" is defined by the specific triangulation that is used (the size of triangles being smaller than robot sensor radius by construction in our algorithm).

*Theorem 2:* If the min-cuts subroutine that is used is correct and complete, and for all $T \in D$ the center of the circumcircle of $T$ is located within $T$, then the method we present is correct and resolution complete

*Proof:* Correctness follows from Theorem 1, the correctness of iterative Delaunay Triangulation, and the correctness of min-cuts. Resolution completeness follows from the completeness of iterative Delaunay Triangulation. If a regular triangular lattice is used instead, then correctness and completeness follow from the correctness and completeness of creating a triangular lattice within a region of finite area. □

Alternatively, if a the stochastic min-cuts algorithm is used (as in our experiments), then our method inherits the usual probabilistic caveats that this entails.

*Theorem 3:* If the min-cuts subroutine that is used is probabilistically complete, then the method we present is correct; it is also resolution complete with probability one, in the limit, as the number of iterations tends toward infinity.

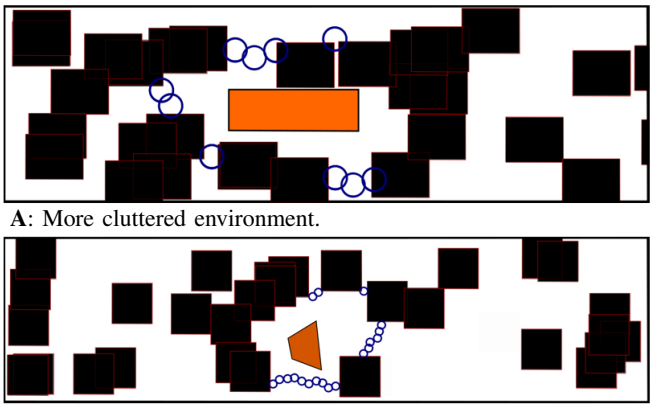
*Proof:* Due to probabilistically completeness, the probability that an optimal min-cuts solution is never found approaches zero, as the number of iterations approaches infinity. □

To summarize, using randomized min-cuts procedure[5] sacrifices completeness, but retains correctness. On the other hand, the benefit of using randomized min-cuts algorithm is that it can be parallelized across the swarm, and tends to find decent solutions in practice—as demonstrated in the next section.

While we have obtained results for completeness and correctness, we are unable, at present, to determine if the algorithm is resolution optimal, even in the case that an optimal min-cuts algorithm is used.

We end this section by discussing a technical detail. Our presentation has assumed that the ground-level projection of free-space is sufficient to understand how adversaries might move between different ground-level patches of free space. If (as has been assumed in our experiments) adversaries cannot climb or enter obstacles and obstacles do not overhang the ground-plane, then this assumption is valid. Otherwise, any routes that connect different parts of the free space must be explicitly added to $\mathcal{X}_{free}$, so that the method can assign agents to guard such passages as necessary.

[5]Karger's algorithm has a nonzero probability of finding the optimal cutting each time it is run, thus a procedure that involves running it over-and-over with different random numbers is probabilistically complete.



**A**: More cluttered environment.

**B**: Less cluttered environment.

Fig. 4. Simulation results showing the placement of agents (blue) to monitor two different $\mathcal{X}_{int}$ (orange) with different $\mathcal{X}_{obs}$ (black) and having different $B$.

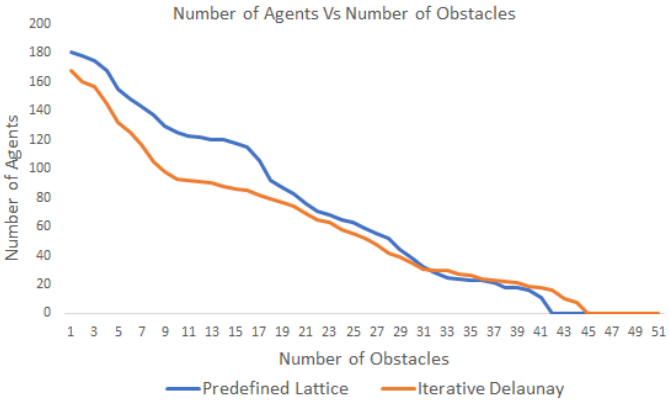



Fig. 5. Plot of Number of Agents vs. number of Obstacle for iterative Delaunay Triangulation and constant lattice over 50 Monte Carlo repeats for each number of obstacles.

## VII. Experiments

### A. Comparison of triangulation methods

In our first experiment we compare the Voronoi-Delaunay method of creating the space portioning cells and motion graph to a modified version of algorithm that uses a regular lattice structure. The lattice structure partitions the map into triangles (note that this results in a Delaunay Triangulation with equidistant points evenly spaced throughout the map such that $B_i$ and $T_i$ have collocated center points for all $i$).

We test both versions on environments containing between 0 and 50 obstacles, where the locations of the obstacles are determined randomly[6]. 50 Monte Carlo trials are run for each method for each number of obstacles. Results reporting the mean number of agents required to isolate a specific $\mathcal{X}_{int}$. All experiments use Karger's Mincut algorithm, which is non-

[6]A point is sampled randomly from the sample space of the map and an obstacle is build around that point

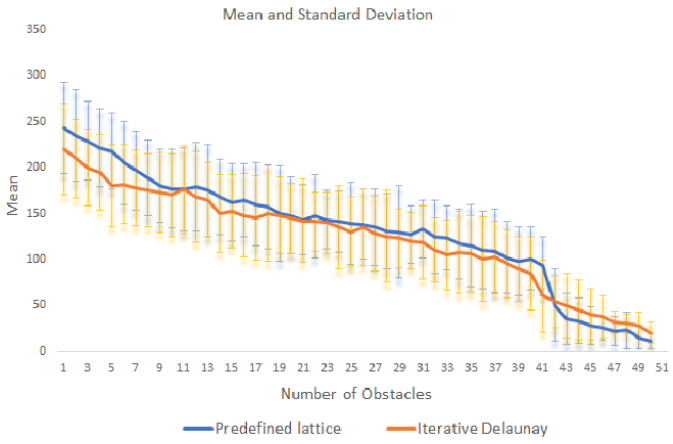



Fig. 6. Plot of Mean and standard deviation for iterative Delaunay Triangulation and constant lattice over 50 Monte Carlo repeats for each number of obstacles.

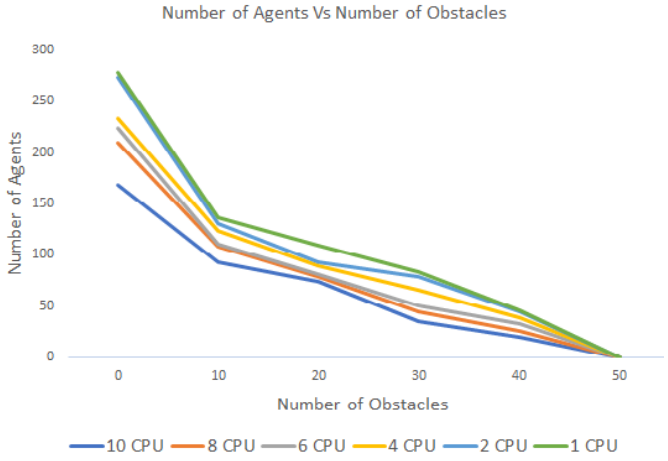



Fig. 7. Plot of Mean Number of Agents vs. Number of Obstacles over 50 Monte Carlo repeats for each number of obstacles. Note that each CPU is located on a different agents.

deterministic (in all cases the Karger's subroutine uses the best solution found in 50 iterations).

### B. Multi-agent parallelization

Karger's Mincut algorithm is both non-deterministic and can easily be parallelized over a multi-robot team by having each agent's CPU run the algorithm with different random seeds. Therefore, in our second experiment we investigate benefits of parallelizing the Karger's Mincut subroutine over the multi-agent system.

We are interested in evaluating the maximum benefit that parallelization may yield, and therefore assume perfect communication between the agents. We perform multiple trials for teams sized from 1 to 10 agents operating in environments with 0 to 50 obstacles. Each run of the experiment involves all agents in the team performing 50 restarts of Karger's Mincut algorithm in parallel. 50 Monte Carlo repeats are performed for each combination of algorithm, team size, and obstacle number. in order to generate performance statistics. A phase consists of 500 iterations and the minimum number of agents after each phase is recorded as shown in Fig.5. The mean results on the number of agents required to monitor for different number of obstacles in the environment is shown in Fig. 7.

## VIII. Results

In general, the algorithm requires fewer agents in environments that contain more obstacles. Note that the number of agents required becomes zero once the region of interest is completely surrounded by obstacles (in which case no paths exists from the map boundary into the region of interest and hence no agent is required to monitor them). Naively covering a particular region of interest boundary requires a constant non-zero number of agents. The trend of decreasing agent requirements as a function of obstacle count demonstrates the method is useful.

The plot in Fig.5 shows that when the number of obstacles were small to moderate (1 to 40 in our experiments) using a Delaunay Triangulation was slightly better, on average, than using a predefined triangular lattice. However, in very cluttered environments (40-50 obstacles) the triangular lattice had better performance. We believe that this happens because the flexibility of iterative Delaunay Triangulation enables the algorithm to adjust the number of triangles according to the shape, size, position and number of obstacles in the map where as predefined lattice fails to do so.

As seen from Fig. 7 the distributed parallelization of Karger's Mincut algorithm across the multi-agent team leads to better performance for the algorithm as the number of agents involved in the computation increases. This result is intuitive — 10 agents with 50 iterations each contribute to 500 restarts of the algorithm, as compared to 2 agents with 100 iterations contributing 200 restarts. Thus, increasing the agents increases the computing power of the system, which in turn provides a better solution in the same amount of (wall-clock) time.

## IX. Conclusions

We present an algorithm that enables a multi-agent team or swarm to efficiently monitor a user specified area of interest against adversarial incursion. The algorithm works by analyzing the geometric structure of the environment, and placing agents at positions in the environment, such as bottlenecks, that collectively surround the region of interest by a combined perimeter of agents and physical obstacles. We believe that the same basic idea can be modified work with other types of environmental models, such as occupancy grids and other space partitioning data structures.

We experimentally demonstrated the usefulness of the algorithm in number of experiments in a simulated urban environment. Moreover, we demonstrated that an important subroutine (Karger's Mincut algorithm) can be parallelized across the multi-agent team, leading to better solutions being calculated, on average, in the same amount of time.

## REFERENCES


[1] D. R. Karger and C. Stein, "An õ (n 2) algorithm for minimum cuts," in *STOC*, vol. 25, 1993, pp. 757–765.

[2] ——, "A new approach to the minimum cut problem," *Journal of the ACM (JACM)*, vol. 43, no. 4, pp. 601–640, 1996.

[3] M. Stoer and F. Wagner, "A simple min-cut algorithm," *Journal of the ACM (JACM)*, vol. 44, no. 4, pp. 585–591, 1997.

[4] F. Katsilieris, M. Lindhé, D. V. Dimarogonas, P. Ögren, and K. H. Johansson, "Demonstration of multi-robot search and secure," in *IEEE ICRA, Anchorage, AK, USA*, 2010.

[5] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro, "Capture of an intruder by mobile agents," in *Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*. ACM, 2002, pp. 200–209.

[6] N. K. Advani, "Decentralized control of an energy constrained heterogeneous swarm for persistent surveillance," *Masters Thesis*, 2017.

[7] C. Trevai, J. Ota, and T. Arai, "Multiple mobile robot surveillance in unknown environments," *Advanced Robotics*, vol. 21, no. 7, pp. 729–749, 2007.

[8] D. Anisi and P. Ögren, "Minimum time multi-ugv surveillance," in *Optimization and Cooperative Control Strategies*. Springer, 2009, pp. 31–45.

[9] A. Kolling and S. Carpin, "The graph-clear problem: definition, theoretical properties and its connections to multirobot aided surveillance," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 1003–1008.

[10] D. Borra, F. Pasqualetti, and F. Bullo, "Continuous graph partitioning for camera network surveillance," *IFAC Proceedings Volumes*, vol. 45, no. 26, pp. 228–233, 2012.

[11] E. Semsch, M. Jakob, D. Pavlicek, and M. Pechoucek, "Autonomous uav surveillance in complex urban environments," in *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 02*. IEEE Computer Society, 2009, pp. 82–85.

[12] L. Geng, Y. Zhang, J. Wang, J. Y. Fuh, and S. Teo, "Mission planning of autonomous uavs for urban surveillance with evolutionary algorithms," in *2013 10th IEEE International Conference on Control and Automation (ICCA)*. IEEE, 2013, pp. 828–833.

[13] S.-W. Ryu, Y.-h. Lee, T.-Y. Kuc, S.-H. Ji, and Y.-S. Moon, "A search and coverage algorithm for mobile robot," in *2011 8Th international conference on ubiquitous robots and ambient intelligence (URAI)*. IEEE, 2011, pp. 815–821.

[14] S. A. Sadat, J. Wawerla, and R. Vaughan, "Fractal trajectories for online non-uniform aerial coverage," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2971–2976.

[15] P. DeLima and D. Pack, "Maximizing search coverage using future path projection for cooperative multiple uavs with limited communication ranges," in *Optimization and Cooperative Control Strategies*. Springer, 2009, pp. 103–117.

[16] L. Wu, M. Á. G. García, D. P. Valls, and A. S. Ribalta, "Voronoi-based space partitioning for coordinated multi-robot exploration," *Journal of Physical Agents*, vol. 1, no. 1, pp. 37–44, 2007.

[17] G. D. Kazazakis and A. A. Argyros, "Fast positioning of limited-visibility guards for the inspection of 2d workspaces," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3. IEEE, 2002, pp. 2843–2848.

[18] T. H. Chung, G. A. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics," *Autonomous robots*, vol. 31, no. 4, p. 299, 2011.

[19] S. A. Bortoff, "Path planning for uavs," in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*, vol. 1, no. 6. IEEE, 2000, pp. 364–368.

[20] D. Yershov, M. Otte, and E. Frazzoli, "Fast collision checking: From single robots to multi-robot teams," in *In IEEE International Conference on Robotics and Automation, Workshop on optimal robot motion planning (WORMP)*, 2015.

[21] L. Ma, L. Cheng, W. Han, L. Zhong, and M. Li, "Cultivated land information extraction from high-resolution unmanned aerial vehicle imagery data," *Journal of Applied Remote Sensing*, vol. 8, no. 1, p. 083673, 2014.

[22] R. Zhu, D. Sun, and Z. Zhou, "Cooperation strategy of unmanned air vehicles for multitarget interception," *Journal of guidance, control, and dynamics*, vol. 28, no. 5, pp. 1068–1072, 2005.

[23] Y. Qu and Q. Tian, "Multi-uav cooperative positioning based on delaunay triangulation," in *2010 International Conference on Computational Aspects of Social Networks*. IEEE, 2010, pp. 401–404.

[24] R. Dai and J. Cochran, "Path planning and state estimation for unmanned aerial vehicles in hostile environments," *Journal of guidance, control, and dynamics*, vol. 33, no. 2, pp. 595–601, 2010.

[25] M. A. Mostafavi, C. Gold, and M. Dakowicz, "Delete and insert operations in voronoi/delaunay methods and applications," *Computers & Geosciences*, vol. 29, no. 4, pp. 523–530, 2003.