2014 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS 2014)
September 14-18, 2014, Chicago, IL, USA

Geodesic Topological Voronoi Tessellations in Triangulated Environments with Multi-Robot Systems

Seoung Kyou Lee[1], Sándor P. Fekete[2] and James McLurkin[1]

Abstract— **Positioning a group of robots at the center of their *geodesic voronoi cells* minimizes the worst-case response time for any robot to arrive at an exogenous event in the workspace. We construct these cells in a distributed fashion, building on our prior work on triangulating unknown spaces with multi-robot systems. This produces a *physical data structure* – a set of triangles formed by the positions of the robots that can be used to perform coverage control. This paper presents: 1) A discrete approximation of the geodesic Voronoi cell using the multi-robot triangulation. We call this a *topological Voronoi cell*, and show that it can be computed efficiently in a distributed fashion and with theoretical guarantees compared to continuous version. 2) A local motion controller to guide navigating robots to the centroid of their topological Voronoi cell. This controller uses bounded communications with a fixed constant, but can produce local extrema that trap navigating robots away from the optimal position. 3) An enhanced local controller using *navigation agents* to help guide the navigating robot to the optimal position in its Voronoi cell. It also uses bounded communications, but with a constant that can be tuned to trade communications bandwidth for increased accuracy. 4) Hardware experiments that compute the topological Voronoi cell on a group of 14 robots, simulation results that demonstrate local extrema, and the effectiveness of the virtual navigation agents, and simulation results comparing the performance of the patrolling algorithm using and not using topological Voronoi cells.**

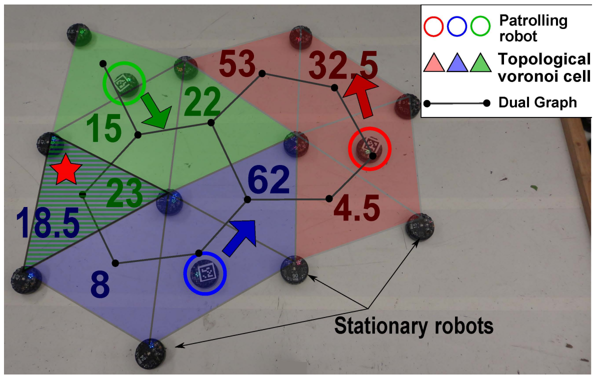


Fig. 1: Sample geodesic Voronoi tessellation of triangulated network with simple patrolling test (colored circles). Black dots and lines indicate the dual graph of the triangulation. Triangles including each patrolling robot become source vertices in the dual graph and builds a BFS tree. Other non-source triangles then belong to the topological Voronoi cell whose geodesic hop to corresponding source triangle is minimum. A triangle which has equal geodesic hops from several source triangles can belong to multiple topological Voronoi cells (left most triangle with stripe pattern). The number in each triangle indicates refresh time for simple patrolling algorithm.

I. Introduction and Related Work

Many practical applications of multi-robot systems, such as search-and-rescue, exploration, mapping and surveillance require robots to disperse across a large geographic area. Large populations of robots can maintain coverage of the environment after the dispersion is complete. These large populations will require the individual robots to be low-cost, precluding the use of expensive sensors or actuators. A solution is to deploy a heterogeneous group of robots, using many small low-cost robots to map the environment, and only a few capable robots to respond to events. Therefore, we need to solve a *coverage control* problem: how to position the mobile robots for optimal response to external events.

There are many approaches to coverage control [1], [2], [3], [4]. One approach is to position robots at the center of their *Geodesic Voronoi cells*. These center positions can optimize system response to a user-provided performance metric, *e.g.* to minimize response times to random events.Using a heterogeneous system of robots allows us to deploy a large number of simple robots to explore the environment in a structured way [5]. In this work, the robots *triangulate* the environment, producing a *physical data structure* – a set of triangles formed by the positions of the robots. This physical data structure (PDS) allows triangles to sense, compute, and communicate the information required to guide navigating robots to optimal positions for coverage control. Fig. 1 shows an example experiment. The three circled robots each construct a topological Voronoi cell composed of triangles. They use the dual graph of the triangulation to navigate to the goal location within their Voronoi cell. In this experiment, the goal location is the triangle that was least recently visited, *i.e.* the one with the largest number.

This paper makes four contributions: **(1)** A discrete approximation of the geodesic Voronoi cell using the multi-robot triangulation. We call this a *topological Voronoi cell*, and show that it can be computed efficiently in a distributed fashion, with theoretical guarantees compared to the continuous, full-information, computation. **(2)** A local motion controller that uses a user-provided distance metric to guide navigating robots to the geodesic centroid of their topological Voronoi cell. This controller uses fixed-size communication messages, but is susceptible to local extrema that can trap navigating robots away from the optimal position in their Voronoi cell. **(3)** An enhanced local motion controller that uses *navigation agents* to guide the navigating robot to the

[1]Seoung Kyou Lee and James McLurkin are with the Computer Science Department, Rice University, 6100 Main St, Houston, TX 77005, USA sl28@rice.edu, jmclurkin@rice.edu

[2] S. Fekete are with the Computer Science Department, TU Braunschweig, Braunschweig, Germany. s.fekete@tu-bs.de

optimal position in its Voronoi cell, avoiding local extrema. It also uses fixed-size messages for each agent, but allows for multiple agents. The user can select the number of agents used; more agents reduces the time required to find the optimal position, but uses more communications bandwidth. **(4)** Hardware experiments that compute topological Voronoi cells on a group of 14 robots, simulation results that demonstrate overcoming local extrema with navigation agents, and simulation results on performance of a patrolling algorithm with and without topological Voronoi cells.

We are interested in solutions for large populations of robots, and focus our attention on approaches applicable to small, low-cost robots with limited sensors and capabilities. In this work, we assume that robots do not have a map of the environment, nor the ability to localize themselves relative to the environment geometry, *i.e.* SLAM-style mapping is beyond the capabilities of our platform. We exclude solutions that use centralized control, as the communication and processing constraints do not allow these approaches to scale to large populations. We also do not assume that GPS localization or external communication infrastructure is available, which are limitations present in an unknown indoor environment. We assume that the communication range is much smaller than the size of the environment, so a multi-hop network is required for communication. Finally, we assume that the robots know the geometry of their local communications network. This *local network geometry* provides each robot with relative pose information about its neighboring robots.

We assume that a large group of simple robots has dispersed into the environment and triangulated the environment. If the number of available robots is not bounded a priori, the problem of minimizing their number for covering all of the region is known as the *Minimum Relay Triangulation Problem*(MRTP); if their number is fixed, the objective is to maximize the covered area, which is known as the *Maximum Area Triangulation Problem* (MATP). Both problems have been studied both for the *offline* scenario, in which the region is fully known, and the *online* scenario, where the region is not known in advance [6], [5]. Online MRTP admits a 3-competitive strategy, while the online MATP does not allow a bounded competitive factor: if the region consists of many narrow corridors, we may run out of robots exploring them, and thereby miss a large room that could permit large triangles. We assume the environment has been triangulated using an online MATP algorithm [7]. These triangles form a physical data structure, which we will rely on for computation and message passing.

Related Work

Cortes $et.\ al.$ uses Lloyd's algorithm with a density function around the target region, such as light source, to deploy robots around the target with resting density [1]. This work, however, uses the standard Voronoi diagram, and is only guaranteed to converge when the voronoi regions are convex. Several authors have extended Lloyd's algorithm to operate in non-convex regions, and must deal with the case when the centroid is not within the navigable region, perhaps inside an concavity, or inside an obstacle. Bhattacharya $et.\ al.$ computes the closest point on the boundary of the region to the centroid, then moves the robot there [2]. Breitenmoser $et.\ al.$ presents a similar approach, but uses the tangent bug algorithm to find the closest point [3]. Yun $et.\ al.$ proposes a distributed vertex substitution algorithm to compute Voronoi centroids for multiple robots in non-convex and discrete space [8]. This method, however, only computes a locally optimal configuration. Breitenmoser $et.\ al.$ uses triangle mesh to compute centroids Voronoi tessellation in non-planar surface [9]. They assume that each robot can get the mesh information with in the communication range and share it with neighbor robots, so that the result highly depends on the length of each mesh's edge.

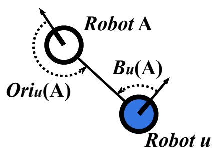
A more rigourous approach is to compute the *geodesic center* of the Voronoi region. Boris presents an algorithm to compute a geodesic Voronoi tessellation in non-convex environment, but does not compute the geodesic center [10]. It is possible to compute the geodesic center in a brute-force fashion by considering each point in the region, but this is computationally intractable for large regions. Many approximations can be found in the robotics literature. Durham $et.\ al.$ describes a pairwise gossip algorithm that converges robots to the center of Voronoi cells constructed on a discrete grid [11]. Pimenta $et.\ al.$ uses objective function with various types of distance including the geodesic distance for geodesic center in non-convex environment and the *power distance* for heterogeneous types of robots [12]. Carlos $et.\ al.$ applies a diffeomorphism that transforms a connected non-convex region to a convex region to run Lloyd's algorithm appropriately [4].

Our results rely on the computational power of many small robots distributed throughout the environment. There is a large body of work on using distributed sensors networks for robot navigation, we cannot cover them all here, but note Batalin's approach, which is similar to our own [13]. Our network is composed of triangles, which provide useful geometric properties. Approaches like Spears $et.\ al.$ in [14] build a triangulated configuration using potential fields, but the network does not have a physical data structure, so the robots never recognize that they form triangles. Our approach allows us to use triangles as computational elements, which support practical distributed computations. Geraerts [15] or Kallmann [16], use a triangulated environment for path planning, but require global information and localization. Our approach is fully distributed, using only local information and communications.

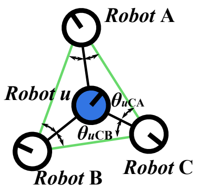
II. Model and Preliminaries

A. Robot Model

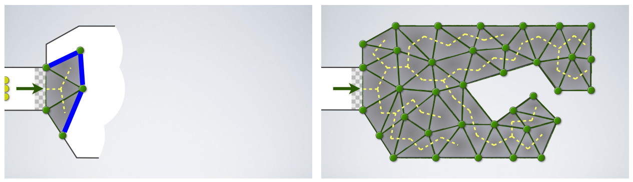
We have a system of n triangulation robots and p navigation robots. The communication network is an undirected graph $G = (V, E)$. Each robot is modeled as a vertex, $u \in V$, where V is the set of all robots and E is the set of all robot-to-robot communication links. The neighbors of each vertex u are the set of robots within line-of-sight communication



(a) Bearing and orientation definitions.



(b) Measuring triangle angles using 2-hop coordinate system.



(c) Robots triangulating an environment.

Fig. 2: **(a)** Robot u can measure the bearing to neighbor A, $B_u(A)$, and the orientation of neighbor A, $Ori_u(A)$. **(b)**, triangle angles (black arrows) are measured from neighbors of robot u, and shared with u using a local broadcast message. **(c)**A large number of robots can explore the environment in a structured way, by triangulating the region as they explore. This produces a *physical data structure* of triangles that can sense, compute, and communicate to run distributed algorithms.

range r_{max} of robot u, denoted $N(u) = \{v \in V \ \mid \{u, v\} \in E\}$. We assume all network edges are also navigable paths. Robot u sits at the origin of its local coordinate system, with the $\hat{x}$-axis aligned with its current heading. Robot u cannot measure distance to its neighbors, but can only measure the *bearing* and *orientation* as shown in Fig. 2a.. We assume that these angular measurements have limited resolution.

Robots share their angle measurements with their neighbors. In this way, robot u can learn of all angles in its 2-hop neighborhood. Fig. 2b shows the relevant *inner angles* of a triangle around u. Each neighbor of u computes these angles from local bearing measurements, then announced them. The communication used by these messages is $O(\max(\delta(u) \in V)^2)$, where $\delta(u)$ is the degree of vertex u.

Each robot has contact sensors that detect collisions with the environment. There is an obstacle avoidance behavior that can effectively maneuver the robot away from these collisions. The robots also have a short-range obstacle sensor that can detect walls closer than $\approx$ 50 cm. The obstacle sensor does not detect neighboring robots.

Algorithm execution occurs in a series of synchronous *rounds*, t_r. This greatly simplifies analysis and is straightforward to implement in a physical system [17]. At the end of each round, every robot u broadcasts a message to all of its neighbors. The robots randomly offset their initial transmission to minimize collisions. During the duration of each round, robot u receives a message from each neighbor $v \in N(u)$. Each message contains a set of public variables, including the sending robot's unique ID number $u.id$. The remaining variables will be defined later, but we note that the number of bits needed for each variable is bounded by $log_2 n$, i.e. the number of bits required to identify each robot. This produces a total message of constant size.

B. Environment Triangulation

Figure 2c shows snapshots of our MATP triangulation algorithm to explore an unknown region [6], [7]. The exploration proceeds in a structured, breadth-first fashion, leaving a triangulated network in its wake. The process of constructing the triangulation and the dual graph builds the physical data structure, and guarantees that adjacent triangles are connected. In this paper, we refer to the triangles as computational elements, but the actual processing and communication occurs on the robots. Our previous work assures that each triangle is "owned" by a robot, and that their is a mapping between the dual graph between triangles and the primal graph between robots [5].

Establishing performance guarantees, both theoretical and practical, relies on the quality of the underlying triangulation. Let r_{max} be the maximum length of a triangulation edge. We also consider a lower bound of r_{min} on the length of the shortest edge in the triangulation; in particular, we assume that the local construction ensures that any non-boundary edge is long enough to let a robot pass between the two robots marking the vertices of the edge, so $r_{\text{min}} \geq 2\delta$, where δ is the diameter of a robot. Finally, angular measurements of neighbor positions let us guarantee a minimum angle of α in all triangles. These constraints give rise to the following:

Definition 2.1: Let $\mathcal{T}$ be a triangulation of a planar region $\mathcal{R}$, with vertex set V. $\mathcal{T}$ is (ρ, α)-*fat*, if it satisfies the following properties:

- The ratio $r_{\text{max}}/r_{\text{min}}$ of longest to shortest edge in $\mathcal{T}$ is bounded by some positive ρ.
- All angles in $\mathcal{T}$ have size at least α.

This definition is used to prove properties of the coverage control based on tessellations.

III. Geodesic Voronoi Diagrams

In this section, we use the triangulation to tessellate the workspace into topological Voronoi cells, which we then use in a patrolling example. Because our triangles are also computational elements in the PDS, we model algorithmic execution on the triangles, and rely on the map between the dual and primal graph to assign the computations to the appropriate owner robots. Our robot sensing does not provide distance information about the graph embedding, only angles. But we can use path length in the dual graph to approximate distances, and then divide the entire triangulated network into n *topologically* equal subnetworks. If we assume that the shape and size of the triangles are similar, we can establish performance guarantees for our results.

A. Topological Voronoi Tessellation

We tessellate the environment around p reference points. Each point is a patrolling robot, u, and is the reference point of its topological Voronoi cell, $\mathcal{V}_u$. The extent of each cell is the set of closest triangles using geodesic distance. In the implementation, a triangle containing a patrolling robot broadcasts a *cell message*. This message disperses within the graph in a breadth-first fashion, building a tree as it propagates. Triangles without robots join the cell whose

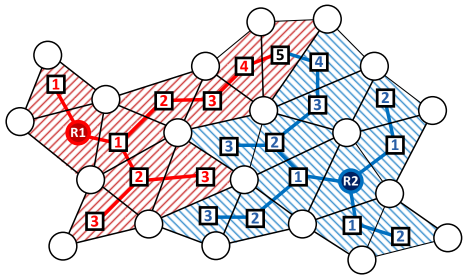


Fig. 3: An example of topological Voronoi tessellation. The red and blue robots are the reference points for the tessellation. The triangles that contain them are sources of BFS broadcast trees. The number in each square indicates the hops in the dual graph to the closest source triangle. Triangles join the cell that is closest, those equidistant join both cells, such as the triangle with 5 hops.

source is closest in the dual graph. Triangles equidistant from multiple robots join each cell. This uses a message of size $O(1)$ and takes $O(D(G))$ rounds of computation, where $D(G)$ is the diameter of the grpah, G. The resulting *topological tessellation* is shown in Fig. 3.

B. Covered Area

In this section, we establish some properties of (ρ, α)-fat triangulations, using the definitions from Section II. See our previous paper [5] for more detailed proofs.

Theorem 3.1: Consider a (ρ, α)-fat triangulation of a set V with n vertices, with maximum edge length r_{max} and minimum edge length r_{min}. Then the area of any two triangles is within $\sqrt{3}\rho^2/2sin(\alpha)$ of each other.

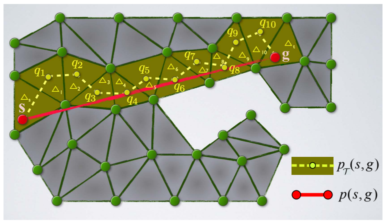
Note that in a practical setting, ρ will be much smaller than the theoretically possible worst case.

C. Path Stretch within a Region

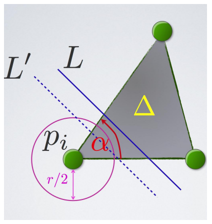
Since we assume network connections are navigable, distances in the dual graph are a discrete approximation of the actual geodesic distance between triangles. We show that with high-quality triangles, this distance can be within a constant factor of the actual distance. Now we establish that the dual graph of our triangulations can be exploited for provably good routing *within one cell of a tesselation*. We include technical details necessary for establishing the main result of Corollary 3.1. We make use of the following terminology.

Definition 3.1: Consider a triangulation $\mathcal{T}$ of a planar region $\mathcal{R}$, with vertex set V. Let s, g be points in $\mathcal{R}$ and let $p(s, g)$ be a polygonal path in $\mathcal{R}$ that connects s to g; let $d_p(s, g)$ be its length. Let Δ_s and Δ_g be the triangles containing s and g, respectively, and let $D(s, g) := \Delta_s, \Delta_1, \ldots, \Delta_\ell, \Delta_g$ be a shortest path in the dual graph of $\mathcal{T}$. Then a $\mathcal{T}$*-greedy* path between s and g is a path $s, q_1, \ldots, q_\ell, g$, such that $q_i \in \Delta_i$, and consecutive vertices of the path are connected by a straight line.

In other words, a $\mathcal{T}$-greedy path between s and g builds a short connection in the dual graph of the triangulation, and then goes from triangle to triangle along straight segments. Note that we do not make any assumptions whatsoever concerning where we visit each of the triangles.



(a)



(b)

Fig. 4: **(a)** A shortest s,g-path (shown in red) in a region covered by a triangulation $\mathcal{T}$. The resulting $\mathcal{T}$-greedy path is depicted in yellow; a shortest dual path is indicated by colored triangles. Note that each point q_i may be *anywhere* in the respective triangle Δ_i. **(b)** A triangle Δ is intersected by a straight line L. If L passes the triangle not too close to one of the endpoints, the length of the intersection is long. If the line passes the triangle close to one of the endpoints (indicated by the dashed line L'), then the intersection with a circle of radius $r_{min}/2$ must be long.

As argued in our previous paper citelsk2014, this yields a bound on the stretch factor of resulting paths.

Theorem 3.2: Consider a (ρ, α)-fat triangulation $\mathcal{T}$ of a planar region $\mathcal{R}$, with vertex set V, maximum and minimum edge length r_{max} and r_{min}, respectively. Let s, g be points in $\mathcal{R}$ that are separated by at least one triangle, i.e., the triangles Δ_s, Δ_g in $\mathcal{T}$ that contain s and g do not share a vertex. Let $p(s, g)$ be a shortest polygonal path in $\mathcal{R}$ that connects s with g, and let $d_p(s, g)$ be its length. Let $p_{\mathcal{T}}(s, g)$ be a $\mathcal{T}$-greedy path between s and g, of length $d_{p_{\mathcal{T}}}(s, g)$. Then $d_{p_{\mathcal{T}}}(s, g) \leq c \cdot d_p(s, g) + 2$, for $c = \lfloor \frac{2\pi}{\alpha} \rfloor \frac{\rho}{\sin(\alpha/2)}$, and $d_{p_{\mathcal{T}}}(s, g) \leq c' \cdot d_p(s, g)$, for $c' = \lfloor \frac{6\pi}{\alpha} \rfloor \frac{\rho}{\sin(\alpha/2)}$.

This provides constant stretch factors in the presence of a single robot. Next, we describe how to use triangulations for tessellation, and (with Corollary 3.1) provide constant stretch factors even in the presence of multiple robots.

D. Global Path Stretch

Finally, we provide performance guarantees for the path stretch *across different cells*, i.e., we show that the topological Voronoi tessellation yields travel distances for the navigation robots using local information that are within constant factors of the shortest distance under full information, even if we end up using a navigation robot that is not closest to a target. The proof is largely based on Theorem 3.2; again we do not have to worry about navigation between locations that are in triangles sharing a vertex: in that case, local navigation provides good results.

Corollary 3.1: Consider a (ρ, α)-fat triangulation $\mathcal{T}$ of a planar region $\mathcal{R}$, with vertex set V, maximum and minimum edge length r_{max} and r_{min}, respectively. Let s_1 be the location of a navigation robot, and g be a target location in $\mathcal{R}$ that belongs to the topological Voronoi cell of s_1. Let s_1 and g be separated by at least one triangle, i.e., the triangles Δ_{s_1}, Δ_g in $\mathcal{T}$ that contain s_1 and g do not share a vertex. Then using a $\mathcal{T}$-greedy path based on a dual Voronoi tessellation is within a constant factor of a shortest geometric path from any navigation robot to g.

Proof: By assumption, Δ_{s_1} is the triangle with minimum hop count in the dual graph. On the other hand, suppose s_2 is the location of a navigation robot that is closest to g in geometry. If $s_2 = s_1$, Theorem 3.2 provides a proof, so suppose s_2 has a larger hop count in the dual graph, meaning it does not get g within its cell. Then we can proceed completely analogous to Theorem 3.2, but use the lower bound on $d_p(s_2, g)$ and the upper bound on $d_{p\mathcal{T}}(s_1, g)$ in order to get $d_{p\mathcal{T}}(s_1, g) \leq c \cdot d_p(s_2, g) + 2$, for $c = \lfloor \frac{2\pi}{\alpha} \rfloor \frac{\rho}{\sin(\alpha/2)}$, and $d_{p\mathcal{T}}(s_1, g) \leq c' \cdot d_p(s_2, g)$, for $c' = \lfloor \frac{6\pi}{\alpha} \rfloor \frac{\rho}{\sin(\alpha/2)}$. ■

This provides constant stretch factors under minimal, conservative assumptions. The practical performance in real-world settings (where the greedy paths do not visit worst-case points in the visited triangles) is considerably better.

E. Experiment: Distributed Patrolling

We use the basic triangulation to support a simple distributed patrolling algorithm. Each triangle stores the time elapsed since its last visit, the *refresh time*, $RT_{\Delta_i(t)}$. The goal is to minimize the refresh time of each triangle. We test two algorithms, one that uses the topological Voronoi cells, and one that does not.

The simplest patrolling algorithm (called LRV, for *least recently visited*) simply moves each robot into the adjacent triangle with the largest $RT_{\Delta_i}(t)$. While simple, this policy produces complete coverage, as shown by the following theorem; for clearer presentation, we refer to the dual graph G induced by the triangulation.

Theorem 3.3: Consider a connected graph $G = (V, E)$ with n vertices, patrolled by p robots according to LRV. Then all vertices get visited infinitely often.

Proof: Let $V_\infty \subseteq V$ be the set of all vertices that get visited infinitely often. Because V is finite, V_∞ is nonempty. Suppose that $V_\infty \neq V$, then the robots must stay within the set V_∞ after some finite time t_0. By definition, they keep visiting all vertices in V_∞, so after $t_1 > t_0$, each of them will have been visited *after time* t_0, for a refresh time of $RT < t - t_0$. Conversely, all vertices in $V \setminus V_\infty$ will have a refresh time $RT \geq t - t_0$, because they do not get visited again. Because G is connected, there must be a vertex $w \in V \setminus V_\infty$ that is adjacent to a vertex $v \in V_\infty$. By definition, v will be visited again; when this happens, the LRV policy ensures that the visiting robot must prefer vertex w (with refresh time at least $t - t_0$) to all vertices in V_∞ (with refresh times less than $t - t_0$), a contradiction to the assumption that w is not visited again. We conclude that $V = V_\infty$. ■

Fig. 5a shows data from this basic patrolling algorithm from the patrolling experiment shown in Fig 1. The experiment starts with one navigating robot; we add another at 1500 and 3600 seconds. As we deploy more navigating robots, the maximum $RT_{\Delta_i}(t)$ decreases, seen in the red line that shows the 400s moving average.

We can implement a better policy using topological Voronoi cells. Each robot navigates to the triangle in its cell, $\mathcal{V}_u$, with max $RT_{\Delta_i(t)}$. To compute the max refresh time in a cell, the triangles run a leader election algorithm, using the refresh time as the quantity[17]. This uses a message of size $O(1)$and takes $O(D(\mathcal{V}_u))$ rounds of computation. Figs 5b-5c show simulation results of patrolling with and without topological Voronoi cells. From [18], we can derive a theoretical lower bound: $RT* = [\frac{\|T\|}{p} - 1]w*$, where $\|T\|$, $w*$, and p are the total number of triangles, the minimum weight of each edge, and the number of patrolling robots, respectively. In our topological tessellation, $w*$ is the optimal transition path to any adjacent triangle, $w* = \frac{1}{6}2\pi r$, which is the distance to travel in an arc between two equilateral triangles around a stationary robot of radius r: $RT* = \frac{\|T\|}{p} \cdot \frac{\pi r}{3}$. Our dual-graph patrolling is directly comparable to many graph patrolling algorithms [18], [19], with the exception that our paths are stretched by the dual graph navigation.

IV. Centers of Non-Convex Voronoi Cells

Much of the robotics work in configuration control uses some variant of the weighted centroid of a region. Given the boundaries of the region, the centroid is straightforward to compute, and guaranteed to be in the interior of any convex region. However, for non-convex regions, the centroid might not be inside, so approximations are often used [3], [2]. For applications that require navigation robots to respond to an event inside its region, the response time is determined by the geodesic distance to the event, so positioning the robot with a centrality metric that uses this distance yields optimal performance. This section uses the distances in the dual graph of the triangulation to compute a *topological center* of a non-convex Voronoi cell.

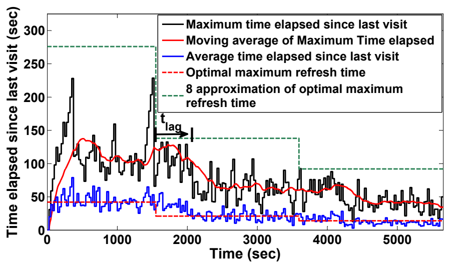
There are many types of topological centrality measures (at least 9, by our count), the choice depends on the application. If we desire a fast response time for a exogeneous event in a region, the *closeness centrality* and the *eccentricity centrality* (both in [20]) minimize the average and worst-case response time, respectively. In this paper, we use the eccentricity centrality (EC) to provide the min-max response time, but our technique can apply to any centrality metric that can be computed using fixed-size messages. For a given vertex in the dual graph u (or its triangle, Δ), and a function to compute the shortest hop count between two vertices in the dual graph, $d_s(u, v)$ the value of the EC metric is:

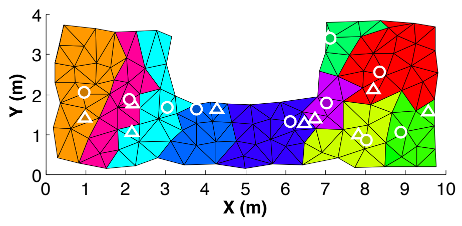
$$EC(\Delta) \equiv EC(u) = \frac{1}{\max_{v \in V} d_s(u, v)} \quad (1)$$

A. Robot Navigation to Center

We can compute the center of a cell in a brute-force fashion if *each* triangle builds a BFS tree, computes its max distance, then selects the triangle with the min-max in the cell. The entire environment contains $|T|$ triangles, requiring $|T|$ broadcast trees, and therefore messages of size $O(|T|)$. Since $|T| \geq O(n/3)$, these are impractically-sized messages.

Our solution is to compute the center iteratively, while navigating to it. Our motion controller only considers the three triangles adjacent to the one occupied by the navigating robot, triangle Δ, shown in Fig. 6a. Given a metric, $m \in C_3(u)$, each triangle adjacent to Δ broadcasts a BFS tree in the dual graph, then runs a convergecast to collect metric values. These three trees do not overlap, so the messages required are only $O(1)$ broadcast+convergecast messages.


(a)


(b)

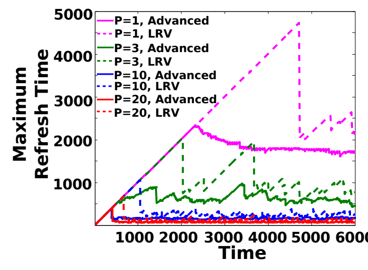

(c)

Fig. 5: **(a)** Patrolling experiment result in triangulation shown in Fig. 1.It starts from one patrolling robot, and adds one more patrolling robot after 1500 seconds and 3600 seconds.**(b)** Distributed patrolling simulation screen shot (with an advanced policy). We test with 1, 3, 10, and 20 patrolling robots. Circles and equilateral triangles represent robots and the target triangle of each robot, respectively. Each color represents each topological Voronoi cell. **(c)** Comparison of the maximum refresh time. Dashed and Solid lines represent the maximum refresh time result with the LRV and advanced policy using topological Voronoi cells, respectively.

The robot in Δ moves toward the adjacent triangle that has the lowest metric value. Unfortunately, many metrics, including EC, do not produce a unique destination within a cell with this type of local controller, and can trap the navigating robots in a local minima. Figs. 6c and 6d shows an example of local minimum in A-shaped and rectangle-shaped environment. We deal with local minima in Sec.IV-B.

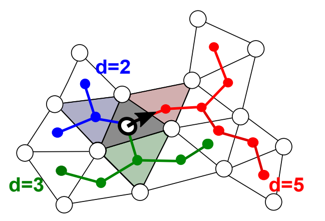
Algorithm 1 shows this itterative approach to guide a patrolling robot to the center of its topological Voronoi cell. Lines 2-9 evaluate the centrality metric on adjacent triangles. Line 11 moves the navigating robot towards the triangle with the smallest value of the metric. The process repeats until the robot is in the triangle that is the center of the cell.

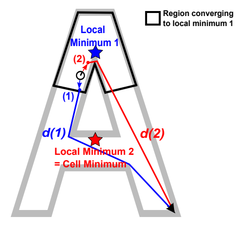
Algorithm 1 REALAGENTS

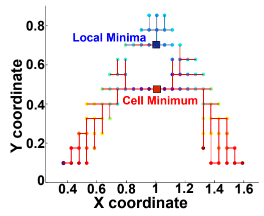
```
1:  while TRUE do
2:      $\Delta_{adj} \leftarrow$ GETADJTRIANGLES()
3:      $\Delta_{goal} \leftarrow \emptyset$
4:      $valMin \leftarrow \infty$
5:      for all $\Delta_i \in \Delta_{adj}$ do
6:          $val \leftarrow EC(\Delta_i)$
7:          if $val < valMin$ then
8:              $\Delta_{goal} \leftarrow \Delta_i$
9:              $valMin \leftarrow val$
10:     if $\Delta_{goal} \neq \emptyset$ then
11:         $MoveRealAgentToTargetTri(\Delta_{goal})$
```

B. Robot Navigation to Center with Agents

To address the local minima problem, we use a probabilistic approach. Algorithm 2 briefly states an algorithm for the approach. Here we refer to an owner of a triangle as just a triangle. With probability p, each triangle in the robot's cell creates a *navigation agent*. Each of these agents broadcasts a message throughout the cell of the form {agentID, agentMetricValue, messageHops}. There are k slots for broadcast messages, a triangle only creates an agent if there is a free slot. The navigation agents follow the same motion policy as the navigation robots, but propagate at network speeds. The physical robot follows the broadcast message from the agent with the best *agentMetricValue*. If multiple agents reach the same minima, one remains; the others are destroyed. An agent that remains stationary for more than $2D(\mathcal{V}_u)$ rounds, while not having the best metric value in $\mathcal{V}_u$ is destroyed. This process continues indefinitely, with one agent remaining at the best value found so far, and $k-1$ agents in motion. Eventually, a virtual agent will be created in the basin of the cell's global minima, propagate there, and remain indefinitely. Increasing the number of agents reduces the time required to find the best value in $\mathcal{V}_u$, but adds communications cost. Fig. 7 shows simulation runs with


(a)


(b)


(c)

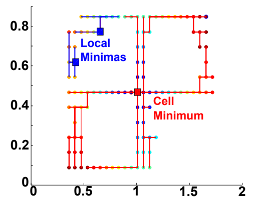

(d)

Fig. 6: **(a)** Distributed algorithm to choose the triangle having the min-max geodesic hops among all adjacent triangles in same topological Voronoi cell. The robot in the gray triangle builds a BFS tree rooted at each adjacent triangle and computes the min-max hop among these three sub-trees. The min-max hop in the red, blue, and green sub-trees are 5, 2, and 3, respectively. Therefore, the robot moves into the red adjacent triangle. **(b)** An A-shaped environment where convergence to the local minima (blue star) happens. The agent moves toward the adjacent point at (2) because its min-max distance, $d(2)$, is less than other min-max distance, such as at $d(1)$. **(c)-(d)** Simulation result that shows the all converging paths from each starting vertices to the local minima or the cell minimum in the dual graph of A-shaped and rectangle-shaped environment. Both of them contain at least one local minima.

Algorithm 2 VIRTUALAGENTS

```
1: Initially, each owner that has the physical data structure of its triangle randomly generates a message which stands for a virtual agent in the triangle with probability, $p$. An owner robot of triangle $T_i$ then runs below algorithm.
2: while ISREALAGENTINCENTROID() = $FALSE$ do
3:     $T_{adj} \leftarrow$ GETADJTRIANGLES()
4:     $maxHop \leftarrow \infty$
5:     $T_{max} \leftarrow \emptyset$
6:     for all $T_j \in \mathcal{T}_{adj}$ do
7:         $h_j \leftarrow$ GETMINMAXHOP($T_j$)
8:         if $h_j < maxHop$ then
9:             $T_{max} \leftarrow T_j$
10:            $maxHop \leftarrow h_j$
11:    if $T_{max} = \emptyset$ then
12:        ELECTCELLMIN()
13:        if $IsCellMin(T_i)$ then
14:            $GuideRealRobotToTri(T_i)$
15:    else
16:        $MoveVirtualAgentToTargetTri(T_{max})$
```

virtual agents guiding robots to the cell's global minima.

Each agent has the same communications requirements as the navigating robot; a broadcast+convergecast message to find the adjacent triangle with the deepest tree in the cell. Therefore the total communications complexity for each cell is $O(k+1)$. Note that these messages do not travel outside of the cell, so the total global message complexity for the entire system is also $O(k+1)$ – it does not depend on the number of cells(*i.e.* the number of robots). Running time requires worst-case $O(k \cdot D(G))$ rounds to select the initial k agents. It takes $O(D(G))$ rounds for an agent to travel the entire region, and $O(D(G))$ rounds for a new agent to be created when two reach the same triangle.

We can trade-off between message size and expected execution time. The expected time to start an agent in the basin of the cell's global minima is governed by the Bernoulli trials. Let F denote an event that k virtual agents converging to local minima are generated by triangles in some V_u. The probability of the event F, $P(F) = \frac{c(|V_u|)(c(|V_u|)-1)\ldots(c(|V_u|)-k+1)}{|V_u|(|V_u|-1)\ldots(|V_u|-k+1)}$, where $|V_u|$ is the total number of triangles in cell V_u, and $c(|V_u|)$ are the number of triangles in the subset of V_u that eventually converge to local minima. The expected number of trials to achieve a success is given by $\frac{1}{1-P(F)}$. The maximum lifespan of an agent is $O(D(G))$ rounds, so the expected execution time becomes $\frac{O(D(G))}{1-P(F)}$. This shows the relationship between communications bandwidth (the number of agents, k) and execution time.

C. Experiment: Navigation to Topological Center

To test navigation to the topological center, we generated 25 cases of $(3.81, 0.31rad)$-*fat* triangulations. Fig. 8 shows these triangulations with two robots (in red) which create two topological Voronoi cells. Each cell uses two virtual

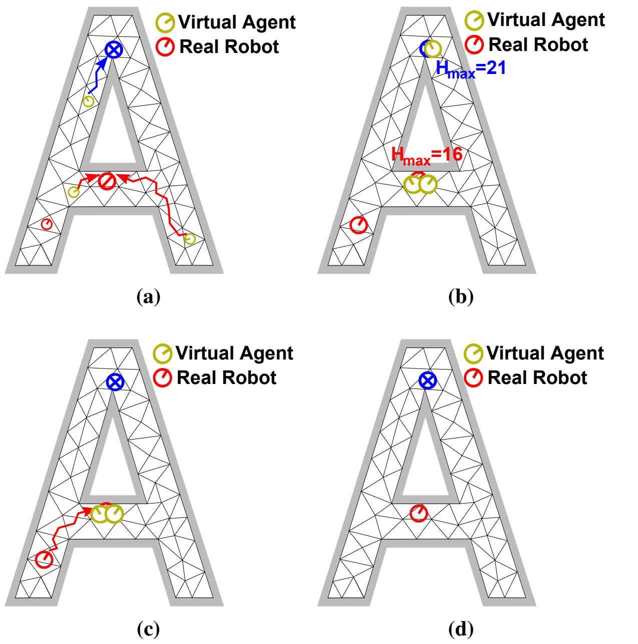


Fig. 7: Illustrations of virtual agent method. **(a)**: Each cell generates k virtual agents. Each of them continuously moves toward the min-max hop adjacent triangle until it reaches local minimum triangle. **(b)**: Each agent is also broadcasting its current min-max value to elect the minimum value among all agents in the cell. **(c)**: A triangle which is the minimum min-max hop then starts to build a BFS tree in corresponding dual graph of the cell. A robot moves toward the triangle using local controller until it reaches the global minima.

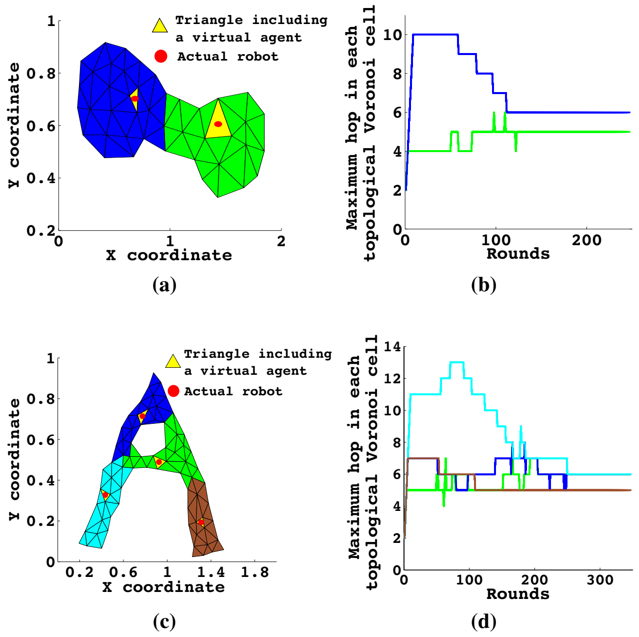


Fig. 8: (Left) Resulting Voronoi tessellation in dumbbell-shaped environment (Fig. **(a)**) and A-shaped environment(Fig. **(c)**). Yellow triangles and red dots indicate triangles containing virtual agents and actual robots, respectively. (Right) Change of maximum hop of each topological Voronoi cell at every round. As actual robots move toward the desired centroids, the maximum hop graphs of each cell converge to the similar value with each other.

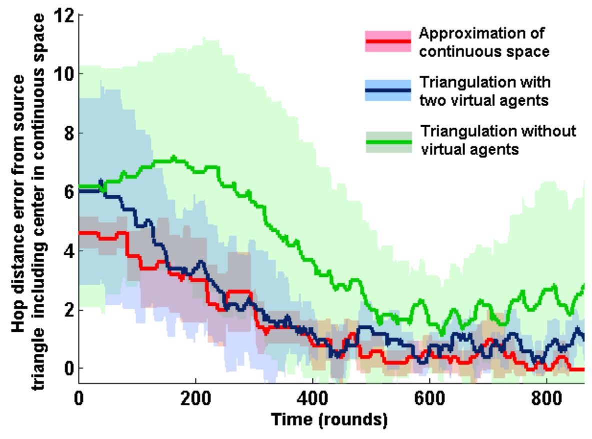


Fig. 9: A plot that shows an error distance (in hop) between the triangle containing the geodesic center in continuous space and the triangle containing an actual robot at every round in A-shaped environment. Red, blue, and green line indicates the hop error in case of algorithm with two virtual agents in approximately continuous space, two virtual agents in triangulation, and no virtual agents in triangulation, respectively.

agents. As the robots move to optimal locations, the value of the EC metric converges to the same value for both robots. We compared topological tessellation with and without navigation agents. Fig. 9 shows the results of 6 trials, compared to a high-resolution discretization of the environment to approximate the continuous solution. The trials without navigation agents often got stuck in local minima, producing a high variance and larger mean. The navigation agents nearly eliminated this problem, producing results almost as good as the ideal solution. These results are difficult to directly compare to existing work. The closest work, Bullo et. al. [18], uses global knowledge of the environment geometry, global localization, long-range communications, and square tessellations of the workspace. Our approach uses a triangular tessellation, is fully distributed, uses local communications, and no shared global knowledge. However, in the limit of arbitrarily small tessellation elements, our two approaches will both converge to the geodesic center.

V. Conclusion

We demonstrated the utility of triangulations for distributed computation. Our construction of the topological Voronoi cell is computationally efficient, and useful for many applications. Navigating to centers of non-convex regions is a common technique for configuration control, and using a probabilistically complete approach allows bounded communications. Both techniques compare well to optimal results.

VI. Acknowledgements

This work was supported by the National Science Foundation under CPS-1035716 and CNS-1330085.

References

[1] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *Robotics and Automation, IEEE Transactions on*, vol. 20, no. 2, pp. 243–255, 2004.

[2] S. Bhattacharya, N. Michael, and V. Kumar, "Distributed coverage and exploration in unknown non-convex environments," in *Distributed Autonomous Robotic Systems*. Springer, 2013, pp. 61–75.

[3] A. Breitenmoser, M. Schwager, J.-C. Metzger, R. Siegwart, and D. Rus, "Voronoi coverage of non-convex environments with a group of networked robots," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 4982–4989.

[4] C. Caicedo-Nuez and M. Zefran, "A coverage algorithm for a class of non-convex regions," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*. IEEE, 2008, pp. 4244–4249.

[5] S. K. Lee, S. P. Fekete, and J. McLurkin, "Geodesic topological voronoi tessellations in triangulated environments with multi-robot systems," in *Robotics and Automation, 2014. ICRA 2014. IEEE International Conference on*. IEEE, 2014.

[6] S. P. Fekete, T. Kamphans, A. Kröller, J. Mitchell, and C. Schmidt, "Exploring and triangulating a region by a swarm of robots," in *Proc. APPROX 2011*, ser. LNCS, vol. 6845. Springer, 2011, pp. 206–217.

[7] A. Becker, S. P. Fekete, A. Kröller, S. K. Lee, J. McLurkin, and C. Schmidt, "Triangulating unknown environments using robot swarms," in *Proc. 29th Annu. ACM Sympos. Comput. Geom.*, 2013, pp. 345–346, video available at http://imaginary.org/film/triangulating-unknown-environments-using-robot-swarms.

[8] S.-k. Yun and D. Rusy, "Distributed coverage with mobile robots on a graph: Locational optimization," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 634–641.

[9] A. Breitenmoser, J.-C. Metzger, R. Siegwart, and D. Rus, "Distributed coverage control on surfaces in 3d space," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 5569–5576.

[10] B. Aronov, "On the geodesic voronoi diagram of point sites in a simple polygon," *Algorithmica*, vol. 4, no. 1-4, pp. 109–140, 1989.

[11] J. W. Durham, R. Carli, P. Frasca, and F. Bullo, "Discrete partitioning and coverage control for gossiping robots," *Robotics, IEEE Transactions on*, vol. 28, no. 2, pp. 364–378, 2012.

[12] L. Pimenta, V. Kumar, R. C. Mesquita, and G. Pereira, "Sensing and coverage for a network of heterogeneous robots," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*. IEEE, 2008, pp. 3947–3952.

[13] M. Batalin and G. S. Sukhatme, "Using a sensor network for distributed Multi-Robot task allocation," in *IEEE International Conference on Robotics and Automation*, New Orleans, Louisiana, Apr. 2004, pp. 158–164.

[14] W. M. Spears, D. F. Spears, J. C. Hamann, and R. Heil, "Distributed, Physics-Based control of swarms of vehicles," *Autonomous Robots*, vol. 17, no. 2, pp. 137–162, 2004.

[15] R. Geraerts, "Planning short paths with clearance using explicit corridors," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1997–2004.

[16] M. Kallmann, "Path planning in triangulations," in *Proceedings of the IJCAI workshop on reasoning, representation, and learning in computer games*, 2005, pp. 49–54.

[17] J. McLurkin, "Analysis and implementation of distributed algorithms for Multi-Robot systems," Ph.D. thesis, Massachusetts Institute of Technology, 2008.

[18] F. Pasqualetti, A. Franchi, and F. Bullo, "On cooperative patrolling: Optimal trajectories, complexity analysis, and approximation algorithms," *IEEE Transactions on Robotics*, vol. 28, no. 3, pp. 592–606, 2012.

[19] Y. Elmaliach, N. Agmon, and G. A. Kaminka, "Multi-robot area patrol under frequency constraints," *Annals of Mathematics and Artificial Intelligence*, vol. 57, no. 3-4, pp. 293–320, 2009.

[20] D. Koschützki, K. A. Lehmann, L. Peeters, S. Richter, D. Tenfelde-Podehl, and O. Zlotowski, "Centrality indices," in *Network analysis*. Springer, 2005, pp. 16–61.