Distributed Voronoi Neighbor Identification from Inter-Robot Distances

Matthew L. Elwin, Randy A. Freeman, and Kevin M. Lynch

***Abstract*—Algorithms for identifying Voronoi neighbors and constructing Voronoi regions are useful in many distributed robotics applications. Existing methods that perform these tasks using only the distances between robots assign coordinates to each potential neighbor before applying another algorithm to find the Voronoi neighbors. Our method finds the Voronoi neighbors more efficiently; identification occurs directly from inter-robot distances, without first assigning coordinates. We prove the algorithm's correctness, analyze its computational complexity, and demonstrate its effectiveness in the presence of noise via simulation with an experimentally validated sensor model.**

***Index Terms*—Sensor networks, distributed robot systems, computational geometry.**

I. Introduction

VORONOI diagrams are useful in many multi-robot applications. When the position of each robot is considered as a generator point for the Voronoi diagram, a given robot's Voronoi region represents the area that is closer to that robot than to any other. This property makes Voronoi diagrams attractive for tasks such as distributed coverage control [1], [2], [3], environmental estimation [4], [5], and querying in spatial databases [6]. Additionally, the average number of Voronoi neighbors per robot is constant and, in the planar case considered here, is at most six [7]. By communicating with only their Voronoi neighbors, robots can maintain network connectivity while reducing the resource usage of distributed algorithms whose memory and computation requirements increase with the number of communication neighbors.

Many algorithms exist for finding Voronoi diagrams (and their dual, Delaunay triangulations). These algorithms assume knowledge of the robot positions and include the incremental, divide and conquer, and plane sweep approaches [7].

In a distributed setting, the algorithm of [8] allows robots in a sensor network to construct their Voronoi region, assuming that every robot knows its own position. When robots cannot directly communicate with all their Voronoi neighbors the methods of [9] and [10] can be employed.

When robots have only relative distance measurements, finding their Voronoi neighbors and regions becomes more difficult. One approach is to use inter-robot distance measurements to localize robots in a coordinate system and then proceed to find Voronoi regions using traditional methods. Extensive reviews of localization are in [11], [12].

In contrast to existing methods, our algorithm allows robots to determine their Voronoi neighbors directly from inter-robot distance measurements, without first performing localization. The algorithm depends on only basic arithmetic operations and terminates after visiting each Voronoi neighbor; thus it can be significantly more efficient than first requiring localization of nearby robots. The algorithm preserves topological relationships between neighbors in the Voronoi graph, allowing robots to detect if they are on the boundary of the convex hull of the group without additional computation. Once identified (and only if desired), the Voronoi region can be described in local coordinates; the topological information from the identification stage makes this step easier than solving a general localization problem and may make localizing non-Voronoi neighbors unnecessary. In local coordinates, the shape of the Voronoi region can be determined, although its location and orientation in the plane will be unknown. The edges of the Voronoi region are associated with the neighbor that induced them. The basic algorithm requires one communication step; adding another communication round allows the robots to compute transformations between their own and their neighbors' regions. If one robot knows its absolute position and orientation, the robots can be placed in a global coordinate system.

We test the algorithm with noisy measurements using the received signal strength indicator (RSSI) of an XBee radio. Simulations with an empirically validated XBee RSSI noise model indicate that error in the identification results is proportional to the measurement noise magnitude.

II. Problem Setup

A. The Robots

Consider $n \geq 3$ robots with unique identifiers $i \in \mathcal{I}$ and positions $p_i \in \mathcal{P} \subset \mathbb{R}^2$, where $\mathcal{I} = 1 \dots n$ and $\mathcal{P} = \{p_1, \dots, p_n\}$ is the set of the robots' positions. The robots communicate with and sense the distance to nearby robots. We model these interactions using the weighted undirected *detection graph* $\mathcal{G}$ and assume that the robots associate incoming measurements and communication packets with the appropriate robot identifier. The vertices of $\mathcal{G}$ correspond to the robots. An edge exists between vertices i and j in $\mathcal{G}$ whenever robots i and j communicate with and sense each other. If an edge exists between vertices i and j, its weight e_{ij} is $(\sigma_{ij}+\sigma_{ji})/2$, where σ_{ij} is robot i's (possibly noisy) distance measurement to robot

Manuscript received September 10, 2016; Revised December 26, 2016; Accepted January 20, 2017.

This paper was recommended for publication by Editor Nak Young Chong upon evaluation of the Associate Editor and Reviewers' comments. This work is supported by the Office of Naval Research, grant N00014-13-1-0331.

The authors are with the McCormick School of Engineering, Northwestern University, Evanston, IL, USA (email: {elwin,freeman,kmlynch}@{u.,eecs.,}northwestern.edu).

Digital Object Identifier (DOI): 10.1109/LRA.2017.2665696

j. If no such edge exists $e_{ij} = \infty$. Robots communicate all their measurements to their detection neighbors; thus all the robots can compute e_{ij} between themselves and their detection neighbors. Let $d_{ij} = \|p_i - p_j\|$ be the Euclidean distance between robots i and j. Then, for noise-free sensors, $e_{ij} = d_{ij}$.

B. Voronoi and Delaunay

Every robot has a *Voronoi region* Ω_i, which contains the points that are closer to it than to any other robot:

$$\Omega_i \equiv \{q\colon \|q - p_i\| \leq \|q - p_j\| \text{ for } j \neq i,\, j \in \mathcal{I},\, q \in \mathbb{R}^2\}.$$

The set of Voronoi regions for all robots is the *Voronoi diagram*, $\mathcal{V}(\mathcal{P}) = \{\Omega_1, \ldots, \Omega_n\}$. The robot positions p_i are the *generator points* of the Voronoi diagram. If two Voronoi regions intersect ($\Omega_i \cap \Omega_j \neq \emptyset$ for $i, j \in \mathcal{I}$) that intersection is a *Voronoi edge* and robots i and j are *Voronoi neighbors*; our algorithm lets robots identify these neighbors using communication and inter-robot distances. The set of robot i's Voronoi neighbors is V_i.

To avoid degenerate cases we assume the following:

Assumption 1. *No four robots in $\mathcal{I}$ lie on the same circle.*

Assumption 2. *No three robots in $\mathcal{I}$ lie on the same line.*

For randomly placed robots with noisy inter-robot distance measurements, these assumptions will be satisfied because circles and lines in $\mathbb{R}^2$ have Lebesgue measure zero.

The *Delaunay triangulation* $\mathcal{D}(\mathcal{P})$ is a triangulation with vertices (called *Delaunay vertices*) located at the robot positions $\mathcal{P}$. An edge of $\mathcal{D}(\mathcal{P})$ (called a *Delaunay edge*) exists between two Delaunay vertices whenever the corresponding robots are Voronoi neighbors. A *Delaunay triangle* is any triangle formed by three Delaunay edges. Thus two Voronoi neighbors $\{i, j\}$ form a Delaunay edge, and three unordered pairs of Voronoi neighbors, $\{i, j\}$, $\{j, k\}$, and $\{k, i\}$ form the edges of Delaunay triangle $\{i, j, k\}$. The three robots composing a Delaunay triangle are *mutual Voronoi neighbors*, as they are Voronoi neighbors of each other. Our assumptions lead to the following property of Delaunay triangulations:

Lemma 1. *Under Assumptions 1 and 2, the Delaunay triangulation is a unique tessellation spanning $\mathcal{P}$.*

Proof. See Property D1 in [7]. □

The circle passing through the vertices $\{p_i, p_j, p_k\}$ of a triangle τ is its *circumcircle* $C(p_i, p_j, p_k)$; the circumcircle's center is the *circumcenter* $c(p_i, p_j, p_k)$ and its radius is the *circumradius* $r(p_i, p_j, p_k)$. We also use $r(d_1, d_2, d_3)$ to be the circumradius of a triangle with side lengths d_1, d_2, and d_3 and $r(A)$ to be the radius of circle A. A circumcircle (or triangle) whose interior contains no generator points (equivalently, robots) is called *empty*.

Lemma 2. *A triangle with vertices in $\mathcal{P}$ is Delaunay if and only if its circumcircle is empty. A triangulation spanning $\mathcal{P}$ is Delaunay if and only if all of its triangles are Delaunay.*

Proof. Properties D1, D5, and D6 in [7]. □

Delaunay edges are either *external* and incident to exactly one Delaunay triangle or *internal* and incident to exactly two Delaunay triangles. The boundary of the convex hull of $\mathcal{P}$ is $\overline{\mathrm{CH}}(\mathcal{P})$. Internal and external edges are related to $\overline{\mathrm{CH}}(\mathcal{P})$:

Lemma 3. *Delaunay edge $\{i, j\}$ is either on $\overline{\mathrm{CH}}(\mathcal{P})$ and external or not on $\overline{\mathrm{CH}}(\mathcal{P})$ and internal.*

Proof. Derived from Property D2 in [7]. □

We now state some classical geometric facts.

Lemma 4. *The circumcenter of a triangle lies at the intersection of the perpendicular bisectors of its sides.*

Lemma 5. *The circumcenter of a triangle τ lies:*

- *outside it, in the positive cone formed by the two sides created by the obtuse angle, if and only if τ is obtuse,*
- *on its longest side's midpoint, if and only if τ is right,*
- *inside τ, if and only if τ is acute.*

Let $U(A, p, q)$ and $u(A, p, q)$ be the major and minor arcs corresponding to chord $\{p, q\}$ of circle A (see Figure 1).

Lemma 6. *Let A and B be non-coincident circles that intersect at points p and q, such that $r(A) \geq r(B)$. Then $U(A, p, q)$ lies outside B and $u(A, p, q)$ lies inside B.*

Proof. Omitted for brevity. □

C. Graphs

The *Delaunay graph* is induced by the Delaunay triangulation: its vertices correspond to the robots and an edge with weight d_{ij} exists between any vertices i and j that are connected by a Delaunay edge. Voronoi neighbors and Delaunay graph neighbors are equivalent.

A *path* in a graph is a sequence of m vertices $\{v_1 \ldots, v_m\}$ ($v_k \in \mathcal{I}$) with no repeated elements except possibly $v_1 = v_m$, such that $\{v_i, v_{i+1}\}$, $i = 1, \ldots, m-1$ is an edge in the graph. Consecutive path vertices are neighbors and the edges between those vertices are on the path. A path *contains* the edges and vertices that are on it. A path's *length* is the sum of the weights of its edges. A *cycle* is a path where $v_1 = v_m$.

A path can be flipped by reversing the order of its elements. A cycle can be rotated by removing its first element and appending its second element to the end. A path and all its flips and rotations form a *path class*, an equivalence class of paths. Every path in a path class contains the same vertices and edges. Therefore, a path class *contains* a vertex or an edge if any (and therefore all) paths in it contain that vertex or edge.

Vertices v_i and v_j are *connected* when a path containing v_i and v_j exists. The distance $d(v_i, v_j)$ between vertices v_i and v_j is the length of the shortest path between the vertices, or infinity if no such path exists. A graph is connected if every vertex is connected to every other vertex.

Let Q be a path class containing edge $\{v_i, v_j\}$. A *subpath* of Q is a path whose elements are a subsequence of a path in Q. The *left subpath* of Q, with respect to vertex pair (v_i, v_j), is the longest subpath of Q that has v_i as its first element and does not contain the edge $\{v_i, v_j\}$ (see Figure 1).

The following lemmas relate to paths in a Delaunay graph.

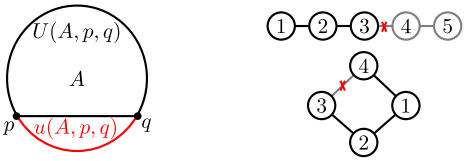

Fig. 1. **Left:** Chord $\{p,q\}$ divides circle A into two arcs: the major arc (longer, black) and the minor arc (shorter, red). **Right:** Left subpath (black) of paths $\{1,2,3,4,5\}$ (top) and $\{1,2,3,4,1\}$ (bottom), induced by pair $(3,4)$. In this example, the left subpath contains all vertices on the original path that, without edge $\{3,4\}$, remain connected to vertex 3.

Lemma 7. *Robot i's Voronoi neighbors V_i can be ordered to form a path in the Delaunay graph. All such* Delaunay paths *belong to the* Delaunay path class*, denoted $\mathcal{D}_i$.*

Proof. The result follows from Lemmas 1 and 3. □

Lemma 8. *Delaunay path class $\mathcal{D}_i$ contains edge $\{j,k\}$ if and only if robots $\{i,j,k\}$ are mutual Voronoi neighbors.*

Proof. Omitted for brevity. □

Lemma 9. *All paths in the Delaunay path class $\mathcal{D}_i$ are cycles if and only if robot i is not on $\overline{\mathrm{CH}}(\mathcal{P})$.*

Proof. The result follows from Lemma 3. □

Lemma 10. *Any cycle that contains only Voronoi neighbors of robot i contains all the Voronoi neighbors of robot i.*

Proof. The result follows from Lemmas 7 and 9. □

D. Metric Space Embeddings

Connected undirected graphs are finite metric spaces: the vertices are the space's elements and the distance between two vertices is the length of the shortest path between them [13]. Let X and Y be metric spaces with respective metrics ρ and σ, and consider a mapping $\eta : X \to Y$. If, for $D \geq 1$, there exists a number $\gamma > 0$ such that

$$\gamma\rho(x,y) \leq \sigma(\eta(x),\eta(y)) \leq \gamma D\rho(x,y) \tag{1}$$

then η is a *D-embedding*. The *distortion* δ of η is the smallest D such that η is a D-embedding. Distortion captures a notion of distance between metric spaces and therefore, between graphs (since connected undirected graphs are metric spaces) [13]. Distortion can also be viewed as the minimum factor such that, after scaling all distances in metric space X by a constant γ, any distance in metric space Y can be achieved by an additional scaling between 1 and δ [13].

Voronoi neighbor identification is an embedding of the metric space associated with the detection graph $\mathcal{G}$ (defined in Section II-A) into the metric space associated with the Delaunay graph. Likewise, localization is an embedding of the metric space associated with $\mathcal{G}$ into Euclidean space [14]. In Section IV-C we use distortion to analyze errors in our Voronoi identification algorithm due to measurement noise.

III. Voronoi Neighbor Identification

Our Voronoi identification algorithm relies on relationships between Voronoi neighbors, circumcircles, and half-planes. Section III-A establishes these relationships and shows how to compute them using only inter-robot distances. Section III-B then uses these geometric results to develop and prove the correctness of our Voronoi identification algorithm.

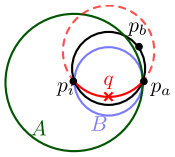

Fig. 2. Theorem 1. All robots except i and a must be outside circle A (green, centered at p_i with radius $r(A)$) because a is the robot closest to i. However, a robot located at point $q \in \mathcal{P}$ cannot be both inside circle $C(p_i,p_a,p_b)$ (black) and outside A when $r(q,p_i,p_a) \geq r(p_i,p_a,p_b)$. For q to be inside $C(p_i,p_a,p_b)$ it must, by Lemma 6, lie on arc $u(C(q,p_i,p_a),p_i,p_a)$ (solid red), which must be inside circle B (blue circle centered at midpoint of p_i and p_a with radius $r(A)/2$). Circle B, however, is always inside circle A.

A. Geometry

We first prove some properties about Voronoi diagrams using only the distances between robots. Let $\mathcal{I}_i = \mathcal{I} \setminus i$.

Lemma 11. *When the robot a is selected such that*

$$a = \arg\min_{a \in \mathcal{I}_i} \|p_i - p_a\| \tag{2}$$

the robots a and i are Voronoi neighbors.

Proof. Property V5 in [7]. □

Lemma 11 shows that robot a, the robot closest to i, is always a Voronoi neighbor of i.[1]

Theorem 1, depicted in Figure 2, lets robot i find a mutual Voronoi neighbor of itself and robot a using the circumradius of a triangle with side lengths d_1, d_2, and d_3, given by

$$r(d_1,d_2,d_3) = \sqrt{\frac{d_1^2 d_2^2 d_3^2}{2d_1^2(d_2^2+d_3^2) - (d_2^2-d_3^2)^2 - d_1^4}}. \tag{3}$$

Theorem 1. *When the robot b is chosen such that*

$$b = \arg\min_{b \in \mathcal{I}_i \setminus a} r(\|p_i - p_a\|, \|p_a - p_b\|, \|p_b - p_i\|) \tag{4}$$

robots i, a, and b are mutual Voronoi neighbors.

Proof. Assume that robots i, a, and b are not mutual Voronoi neighbors. Then triangle $\{i,a,b\}$ is not Delaunay and, by Lemma 2, a robot at position $q \in \mathcal{P}$ must be inside circumcircle $C(p_i,p_a,p_b)$. Figure 2 and omitted details show that no such q exists. Thus triangle $\{i,a,b\}$ is Delaunay and robots i, a, and b are mutual Voronoi neighbors. □

We now introduce some concepts that allow robot i to find additional Voronoi neighbors. Let L_{ij} be the line through robot positions p_i and p_j. This line divides $\mathbb{R}^2$ into two open half-planes: $\mathcal{H}(i,j,k)$, which does not contain the position p_k of robot k and $\widetilde{\mathcal{H}}(i,j,k)$, which contains p_k. By Assumption 2, every robot other than i or j is located in one of these half-planes. Let $H(i,j,k)$ and $\widetilde{H}(i,j,k)$ be the robots located in $\mathcal{H}(i,j,k)$ and $\widetilde{\mathcal{H}}(i,j,k)$, respectively.

Lemma 12 provides conditions under which mutual Voronoi neighbors exist. It refers to Figure 3.

[1] If multiple robots satisfy Equation (2), any of them can be used.

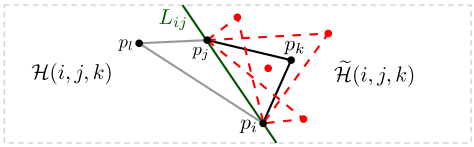

Fig. 3. Lemma 12. When robots i, j, and l are mutual Voronoi neighbors, l must be in $H(i,j,k)$: the red points indicate potential locations for robot l that contradict triangle $\{i,j,k\}$ (black) being Delaunay. If $H(i,j,k)$ contains a robot, then Delaunay edge $\{i,j\}$ does not lie on $\overline{\mathrm{CH}}(\mathcal{P})$ and is one side of two Delaunay triangles. One triangle is $\{i,j,k\}$ and an $l \in H(i,j,k)$ can be chosen such that the other triangle is $\{i,j,l\}$ (gray).

Lemma 12. *Let robots i, j, and k be mutual Voronoi neighbors. Robots i and j have a mutual Voronoi neighbor $l \neq k$ if and only if $H(i,j,k)$ is not empty. Furthermore, if such an l exists, $l \in H(i,j,k)$.*

Proof. Sufficiency: Triangle $\{i,j,l\}$ is Delaunay. If $l \in \widetilde{H}(i,j,k)$ all cases, by Lemmas 1 and 2, lead to contradiction (see Figure 3). Thus, $l \notin \widetilde{H}(i,j,k)$ so $l \in H(i,j,k)$.

Necessity: Robot l exists in $H(i,j,k)$ so p_l is in $\mathcal{H}(i,j,k)$. Since p_k is in $\widetilde{\mathcal{H}}(i,j,k)$, Delaunay edge $\{i,j\}$ is not on $\overline{\mathrm{CH}}(\mathcal{P})$ and, by Lemma 3, it is internal. Therefore, edge $\{i,j\}$ must be incident to two Delaunay triangles. One of these triangles is $\{i,j,k\}$. Denote the other triangle $\{i,j,t\}$. The existence of this triangle implies the existence of a robot $t \neq k$ that is a Voronoi neighbor of robots i and j. The sufficiency argument of this proof implies $t \in H(i,j,k)$. □

Theorem 2, depicted in Figure 4, lets robot i determine a Voronoi neighbor of itself and j, given mutual Voronoi neighbors $\{i,j,k\}$. It must compute the signed circumradius $\hat{r}(\cdot)$ of a triangle in terms of its side lengths:

$$\hat{r}(d_1,d_2,d_3) = \begin{cases} r(d_1,d_2,d_3) & \text{if } \mathrm{obtuse}(d_1,d_2,d_3) = 0 \\ -r(d_1,d_2,d_3) & \text{if } \mathrm{obtuse}(d_1,d_2,d_3) = 1, \end{cases}$$

where

$$\mathrm{obtuse}(d_1,d_2,d_3) = \begin{cases} 1 & d_3 > d_1\,, d_3 > d_2\,, d_1^2 + d_2^2 < d_3^2 \\ 0 & \text{otherwise.} \end{cases}$$

Note that $\mathrm{obtuse}(d_1,d_2,d_3) = 1$ when the angle opposite the side corresponding to d_3 is obtuse.

Theorem 2. *Let robots i, j, and k be mutual Voronoi neighbors and assume that $H(i,j,k)$ is non-empty. Then robots i and j have a mutual Voronoi neighbor $l \neq k$, with*

$$l = \underset{l \in H(i,j,k)}{\arg\min}\ \hat{r}(\|p_i - p_l\|, \|p_j - p_l\|, \|p_i - p_j\|). \tag{5}$$

Proof. Lemma 12 implies that a Voronoi neighbor of i and j exists in $H(i,j,k)$, regardless of $\widetilde{H}(i,j,k)$. We therefore assume, without loss of generality, that $\widetilde{H}(i,j,k) = \{k\}$ and show that the robot l satisfying Equation (5) is a Voronoi neighbor of robots i and j.

Let circle $C(p) = C(p_i,p_j,p)$, circumcenter $c(p) = c(p_i,p_j,p)$, and signed circumradius $\hat{r}(p) = \hat{r}(\|p_i - p\|, \|p_j - p\|, \|p_i - p_j\|)$, for $p \in \mathcal{H}(i,j,k)$. By Lemma 4, $c(p_k)$ and $c(p_l)$ lie on the perpendicular bisector of edge $\{i,j\}$. Therefore, Lemma 5 implies that

$$\hat{r}(p) = \begin{cases} r\left(C(p)\right) & \text{if } c(p) \in \mathcal{H}(i,j,k) \\ -r\left(C(p)\right) & \text{if } c(p) \in \widetilde{\mathcal{H}}(i,j,k) \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

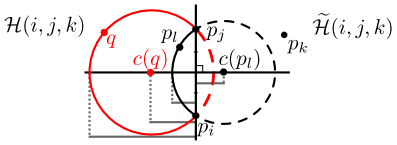

Fig. 4. Theorem 2. When circle $C(p_l)$ (black) has the smallest signed circumradius, any other circumcircle $C(q)$, for a robot located at point $q \in \mathcal{H}(p_i,p_j,p_k)$, must have its circumcenter $c(q)$ to the "left" of circumcenter $c(p_l)$. Therefore, the arc $u(q)$ (solid red) must be to the "left" of arc $u(p_l)$ (solid black). Because q must be on $u(q)$ it can never be inside of $C(p_l)$ and therefore triangle $\{i,j,l\}$ is Delaunay. The horizontal gray lines (from top to bottom) indicate the signed distances $h(p_l)$, $d(p_l)$, $h(q)$, and $d(q)$. In this configuration, $h(p_l)$ is negative.

Let $h(p)$ be the signed perpendicular distance between $c(p)$ and edge $\{i,j\}$, with $h(p) < 0$ when $c(p) \in \widetilde{\mathcal{H}}(i,j,k)$ and $h(p) \geq 0$ otherwise. As $|h(p)|$ decreases, $r(C(p))$ must decrease. When $h(p) > 0$, $\hat{r}(p) = r(p)$ and when $h(p) \leq 0$, $\hat{r}(p) = -r(x)$. Thus, $\hat{r}(p) < \hat{r}(q)$ if and only if $h(p) < h(q)$.

The next three cases show that no robot located at a point $q \in \mathcal{H}(i,j,k)$ is inside $C(p_l)$ when l satisfies Equation (5).

$h(q) < h(p_l)$: In this case $\hat{r}(p_l)$ violates Equation (5).

$h(q) = h(p_l)$: This case violates Assumption 1.

$h(q) > h(p_l)$: Let $u(p)$ be the arc of $C(p)$ that lies in $\mathcal{H}(i,j,k)$ and let $d(p)$ be the perpendicular distance from edge $\{i,j\}$ to the base of $u(p)$. Then q lies on arc $u(q)$ and $d(p)$ decreases with $h(p)$. Thus $d(q) > d(p_l)$, which implies that $u(q)$ (and therefore q) is outside $C(p_l)$ (otherwise $u(q)$ and $u(p_l)$ would intersect). Therefore, $C(p_l)$ is empty. By Lemma 2, triangle $\{i,j,l\}$ is Delaunay and robots i, j, and l are mutual Voronoi neighbors. □

To use Theorem 2, robot i determines which of its detection neighbors are in $H(i,j,k)$, which is equivalent to determining if robot positions p_l and p_k are on opposite sides of the line L_{ij} passing through points p_i and p_j.

To determine whether robots k and l are on the same or opposite sides of L_{ij}, we first define a condition T and two quantities U and W, based on inter-robot distances:

$$\begin{aligned} T &= (d_{ik} \leq d_{jk} \wedge d_{il} \leq d_{jl}) \vee (d_{ik} > d_{jk} \wedge d_{il} > d_{jl}), \\ U &= (f_l - g_l)(f_l + g_l)(f_k - g_k)(f_k + g_k), \\ W &= d_{ij}^2(f_l^2 + f_k^2 + g_l^2 + g_k^2 - d_{ij}^2 - 2d_{lk}^2), \end{aligned}$$

where $f_t = \min(d_{it}, d_{jt})$ and $g_t = \max(d_{it}, d_{jt})$, $t \in \{k,l\}$.

Theorem 3, which refers to Figure 5, uses these quantities to determine if p_k and p_l are on opposite sides of line L_{ij}.

Theorem 3. *Robots k and l are on opposite sides of the line L_{ij} if and only if $S_{ijkl} < 0$, where*

$$S_{ijkl} = \begin{cases} W - U & \text{if } T \\ W + U & \text{otherwise.} \end{cases} \tag{7}$$

Proof. Without loss of generality, we assign local coordinates to the robot locations (see Figure 5).[2] Let $p_i = (0,0)$ and $p_j = (0,d_{ij})$. Let α_t $(t \in \{k,l\})$ be the angle of triangle $\{i,k,t\}$ opposite side $\{i,t\}$ if $d_{it} \leq d_{jt}$ or opposite side $\{j,t\}$ otherwise. That is (from the law of cosines),

$$\alpha_t = \arccos\left(\frac{d_{ij}^2 + g_t^2 - f_t^2}{2d_{ij}g_t}\right). \tag{8}$$

[2]These coordinates are for the proof; the robots need not compute them.

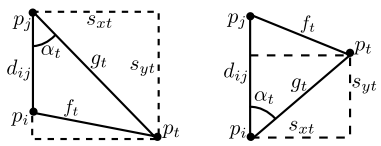

Fig. 5. Temporary coordinate systems for the possible locations of angle α_t relative to triangle $\{i, j, t\}$. Point $p_i = (0, 0)$ and point $p_j = (0, d_{ij})$.

We write the coordinates for points p_k and p_l in terms of:

$$\xi(t, z) = \begin{cases} (zs_{xt}, d_{ij} - s_{yt}) & \text{if } d_{it} \leq d_{jt} \\ (zs_{xt}, s_{yt}) & \text{otherwise} \end{cases}, \tag{9}$$

where $s_{xt} = g_t \sin(\alpha_t)$, $s_{yt} = g_t \cos(\alpha_t)$ and $z \in \{-1, 1\}$. The parameter z determines the side of L_{ij} the point is on.

Fix p_k by letting $p_k = \xi(k, 1)$. We then let

$$p_l = \begin{cases} \xi(l, -1) & \text{if } \|\xi(l, -1) - p_k\| = d_{kl} \\ \xi(l, 1) & \text{if } \|\xi(l, 1) - p_k\| = d_{kl}. \end{cases} \tag{10}$$

Thus, p_l is assigned coordinates consistent with the distance between robots k and l. If $p_l = \xi(l, -1)$, robots k and l are on opposite sides of L_{ij}; otherwise they are on the same side. Let $\Psi = (\|\xi(l, -1) - p_k\|^2 - d_{kl}^2)^2 - (\|\xi(l, 1) - p_k\|^2 - d_{kl}^2)^2$. The condition $\Psi < 0$ determines which branch of Equation (10) is closer to being satisfied. We use this condition because with inexact distances neither equality condition in Equation (10) strictly holds. When $\Psi < 0$, $p_l = \xi(l, -1)$ and robots i and j are on opposite sides of L_{ij}. Substituting Equation (9) into Ψ shows that $\Psi < 0$ and $\Phi < d_{ij}^2$ are equivalent, where

$$\Phi = \begin{cases} s_{xl}^2 + s_{xk}^2 + (s_{yl} - s_{yk})^2 & \text{if } T \\ s_{xl}^2 + s_{xk}^2 + (s_{yl} - s_{yk} - d_{ij})^2 & \text{otherwise.} \end{cases} \tag{11}$$

Calculating s_{xl}, s_{yl}, s_{xk} and s_{yk} in terms of inter-robot distances and substituting into Equation (11) shows that $\Phi < d_{ij}^2$ is logically equivalent to $S_{ijkl} < 0$. □

B. Voronoi Identification Algorithm

The results of Section III-A help us develop a Voronoi identification algorithm and prove its correctness. The sets $I \subset \mathcal{I}$ and $J \subset \mathcal{I}$ are sets of robots and d_{ij} is the distance between robots i and j. The algorithm depends on some basic functions whose implementations we assume are correct:

- CLOSEST-ROBOT(i, I): The robot $j \in I$ that minimizes d_{ij}.
- MIN-RADIUS(i, j, I): The robot $k \in I$ that minimizes $r(d_{ij}, d_{jk}, d_{ik})$.
- MIN-SIGNED-RADIUS(i, j, I): The robot $k \in I$ that minimizes $\hat{r}(d_{ik}, d_{jk}, d_{ij})$
- IN-HALF-PLANE(i, j, k, l): TRUE if $l \in H(i, j, k)$, otherwise FALSE. Implemented in terms of Theorem 3.
- ALL-IN-HALF-PLANE(i, j, k, I): The set $I \cap H(i, j, k)$: all robots $l \in I$ with IN-HALF-PLANE(i, j, k, l) = TRUE.

The above functions are defined using mathematical sets, not data structures such as arrays. Data structure choice has practical implications but does not affect our proofs.

When robots i, j, and k are mutual Voronoi neighbors, NEXT-NEIGHBOR(i, j, k, I) returns either a mutual Voronoi neighbor of robots i and j other than k, if it exists in I, or $\delta \notin \mathcal{I}$, indicating non-existence. The routine implements Lemma 12 (see Algorithm 1). Theorem 4 proves it is correct.

Algorithm 1 Next neighbor implementation

Require: Robots i, j, and k are mutual Voronoi neighbors
Result: A Voronoi neighbor of i and j, or δ if none exists

```
function NEXT-NEIGHBOR(i, j, k, I)
    J ← ALL-IN-HALF-PLANE(i, j, k, I)
    if J = ∅ then
        return δ
    else
        return MIN-SIGNED-RADIUS(i, j, J)
```

Theorem 4. *For all mutual Voronoi neighbors $\{i, j, k\}$ and sets $I \subset \mathcal{I}$, let ν = NEXT-NEIGHBOR(i, j, k, I). Let robot $l \in I$, $l \neq k$ be a Voronoi neighbor of robots i and j, if such an l exists. The following statements hold:*

1) *If l exists then $\nu = l$.*
2) *If $\nu \neq \delta$, then l exists.*
3) *If and only if $\nu = \delta$ then l does not exist.*

Proof. We now prove each condition.

1) If l exists then, by its definition, Theorem 2 implies
$$l = \underset{l \in H(i,j,k)}{\arg\min}\ \hat{r}(d_{il}, d_{jl}, d_{ij}). \tag{12}$$
Therefore $l \in H(i, j, k)$, so $l \in I \cap H(i, j, k)$. Within NEXT-NEIGHBOR, J = ALL-IN-HALF-PLANE(i, j, k, I) = $I \cap H(i, j, k)$, so $l \in J$. Thus ν = MIN-SIGNED-RADIUS(i, j, J) = $\arg\min_{l \in J} \hat{r}(d_{il}, d_{jl}, d_{ij})$. Since $l \in J$, l satisfies Equation (12), and $J \subset H(i, j, k)$, $\nu = l$.
2) If $\nu \neq \delta$, then ν = MIN-SIGNED-RADIUS(i, j, J) and $J \neq \emptyset$. Then J = ALL-IN-HALF-PLANE(i, j, k, I) = $I \cap H(i, j, k)$, so $H(i, j, k) \neq \emptyset$. By Lemma 12, l exists.
3) If $\nu = \delta$, $J = \emptyset$. Since $J = I \cap H(i, j, k)$, and $l \in I$, $l \notin H(i, j, k)$. By Lemma 12, an $l \neq k$ cannot exist.
4) If l does not exist, by Lemma 12, $H(i, j, k) = \emptyset$. Therefore $J = \emptyset$ and $\nu = \delta$. □

The function FIND-VORONOI(i, j, k, W, I), listed in Algorithm 2, finds some Voronoi neighbors of i given mutual Voronoi neighbors $\{i, j, k\}$, the confirmed Voronoi neighbors W, and the potential Voronoi neighbors I. The arguments to FIND-VORONOI(i, j, k, W, I) must meet some conditions:

Condition 1. *For* FIND-VORONOI(i, j, k, W, I)*:* *$W \cap I = \emptyset$ and $V_i \subseteq (W \cup I) \subseteq \mathcal{I}_i$.*

Condition 2. *For* FIND-VORONOI(i, j, k, W, I)*: All elements of W lie on D_i, a subpath of Delaunay path class $\mathcal{D}_i$, with first and second elements $[D_i]_1 = j$ and $[D_i]_2 = k$.*

When FIND-VORONOI(i, j, k, W, I) satisfies Condition 2, $W \subset V_i$, $j \in W$, $k \in W$, $\{i, j, k\}$ are mutual Voronoi neighbors, and $i \notin W$. Essentially, FIND-VORONOI finds Voronoi neighbors along the Delaunay path until it either determines that the path is a cycle or it reaches an endpoint.

Theorem 5. *Assume Conditions 1 and 2 hold for (X, Y, o) =* FIND-VORONOI(i, j, k, W, I)*. Let D_{ijk} be the left subpath of Delaunay path class $\mathcal{D}_i$, from vertex pair (j, k). Then*

Algorithm 2 Find some Voronoi neighbors

Require: Conditions 1 and 2 hold.
Result: A triple (X, Y, o), where
$X \subset V_i$ contains the Voronoi neighbors found,
$Y \subset I$ holds the elements of I that are not in X,
$o =$ Hull if p_i is on $\overline{\mathrm{CH}}(\mathcal{P})$ and $o =$ Interior otherwise.
function FIND-VORONOI(i, j, k, W, I)
  $x \leftarrow$ NEXT-NEIGHBOR(i, j, k, $W \cup I$)
  if $x \in W$ **then**
    return (W, I, Interior)
  else if $x = \delta$ **then**
    return (W, I, Hull)
  else if $x \in I$ **then**
    return FIND-VORONOI(i, x, j, $W \cup \{x\}$, $I \setminus x$)

1) Conditions 1 and 2 hold for FIND-VORONOI(i, j, k, X, Y)
2) $X = D_{ijk} \cup W$.
3) If and only if robot i *lies on* $\overline{\mathrm{CH}}(\mathcal{P})$ *then* $o =$ *Hull.*
4) If $o =$ *Interior then* $X = V_i$.

Proof. By Theorem 4 and Condition 1, the IF in FIND-VORONOI covers all possible x values. We induct on I.

Base: $I = \emptyset$ so $x \notin I$, and there are two cases.

If $x \in W$, by Theorem 4, $x \neq k$ and $\{i, j, x\}$ are mutual Voronoi neighbors. Therefore, by Lemma 8, $\mathcal{D}_i$ contains edge $\{j, x\}$. Under Condition 2, $[D_i]_1 = j$, $[D_i]_2 = k$ and $x \in D_i$: to contain edge $\{j, x\}$, the last element of D_i must be j, so D_i is a cycle. By Lemma 10, $D_i = V_i$, so, by Condition 2, $W = V_i$. Since $X = W$ and $Y = I$, Statements 1, 2 and 4 hold. This case is the only one with $o =$ Interior and, by Lemma 9, i is not on $\overline{\mathrm{CH}}(\mathcal{P})$; therefore, Statement 3 holds.

If $x = \delta$, by Theorem 4, i and j have no mutual Voronoi neighbor other than k; therefore, the only Voronoi neighbor of j in D_i is k. By definition, $[D_{ijk}]_1 = j$ and $[D_{ijk}]_2 \neq k$. By Condition 2, $[D_i]_1 = j$; therefore $[D_{ijk}]_1 = [D_i]_1$. However, because k is the only Voronoi neighbor of j contained in D_i, $D_{ijk} = \{[D_i]_1\}$, so $D_{ijk} \subset W$; therefore, because $X = W$, Statement 2 holds. Also, because $Y = I$, Statement 1 holds. In this case, the only one with $o =$ Hull, Lemma 12 implies $H(i, j, k) = \emptyset$; thus, i is on $\overline{\mathrm{CH}}(\mathcal{P})$ and Statement 3 holds. Statement 4 holds because $o \neq$ Interior.

Inductive: Given the inductive hypothesis that the Theorem is true for all FIND-VORONOI$(i, j, k, W, I \setminus y)$ invocations satisfying Conditions 1 and 2 and for all $y \in I$, we must show that FIND-VORONOI(i, j, k, W, I) is also true. Within FIND-VORONOI, proving the cases when $x \in W$ and $x = \delta$ requires reasoning similar to that of the base case. When $x \in I$ we have $(X, Y, o) =$ FIND-VORONOI$(i, x, j, W \cup \{x\}, I \setminus x)$ and apply the inductive hypothesis to complete the proof. (It can be verified that the arguments to this recursive FIND-VORONOI call satisfy Conditions 1 and 2). □

Algorithm 3 lists VORONOI-NEIGHBORS, which finds all the Voronoi neighbors of i. Figure 6 depicts its operation and Theorem 6 proves its correctness.

Theorem 6. *Let* $(X, Y, o) =$ VORONOI-NEIGHBORS(i, I), *with* $V_i \subseteq I$. *Then the following statements are true:*

Algorithm 3 Find all the Voronoi neighbors of robot i

Require: $V_i \subseteq I \subseteq \mathcal{I}_i$
Result: A triple (X, Y, o) where
$X = V_i$, the Voronoi neighbors of robot i,
$Y \subset I$, the elements in I that are not in V_i,
$o =$ Hull if p_i is on $\overline{\mathrm{CH}}(\mathcal{P})$ and $o =$ Interior otherwise.
function VORONOI-NEIGHBORS(i, I)
  $j \leftarrow$ CLOSEST-ROBOT(i, I)
  $I_1 \leftarrow I \setminus j$
  $k \leftarrow$ MIN-RADIUS(i, j, I_1)
  $I_2 \leftarrow I_1 \setminus k$
  $(X_1, Y_1, o_1) \leftarrow$ FIND-VORONOI(i, j, k, $\{j, k\}$, I_2)
  if o_1 = Hull **then**
    return FIND-VORONOI(i, k, j, X_1, Y_1)
  else if o_1 = Interior **then**
    return (X_1, Y_1, o_1)

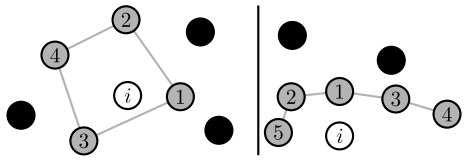

Fig. 6. Algorithm 3. **Both:** Robot i identifies neighbors in the order indicated by the robot identifiers. It uses CLOSEST-ROBOT to find (1), MIN-RADIUS to find (2) and repeatedly uses NEXT-NEIGHBOR to find Voronoi neighbors along the Delaunay path (gray). **Left:** The operation of VORONOI-NEIGHBORS when robot i is not on $\overline{\mathrm{CH}}(\mathcal{P})$. The Delaunay path is a cycle, so when NEXT-NEIGHBOR encounters the previously identified neighbor (2), it has identified all Voronoi neighbors. **Right:** The operation of VORONOI-NEIGHBORS when the robot is on $\overline{\mathrm{CH}}(\mathcal{P})$. When NEXT-NEIGHBOR finds an end of the Delaunay path (4), the search proceeds in the other direction (starting from 2) until encountering the other end of the path (5).

1) $X = V_i$.
2) $o =$ HULL *if and only if robot* i *lies on* $\overline{\mathrm{CH}}(\mathcal{P})$.

Proof. By Lemma 11 and Theorem 1, robots i, j and k are mutual Voronoi neighbors. Conditions 1 and 2 hold for FIND-VORONOI$(i, j, k, \{j, k\}, I_2)$. Applying Theorem 5 shows there are exactly two cases for o_1:

o_1 = Interior: Robot i is not on $\overline{\mathrm{CH}}(\mathcal{P})$ and $X = X_1 = V_i$.
o_1 = Hull: Robot i lies in $\overline{\mathrm{CH}}(\mathcal{P})$ and $X_1 = D_{ijk} \cup \{k\}$. Then, $(X, Y, o) =$ FIND-VORONOI(i, k, j, X_1, Y_1), which satisfies Conditions 1 and 2. We can then apply Theorem 5 to reveal that $X = D_{ikj} \cup D_{ijk} = V_i$ and $o =$ HULL.

In both cases above, $X = V_i$, which proves Statement 1. Analysis shows that $o =$ Hull when robot i is on $\overline{\mathrm{CH}}(\mathcal{P})$ and that $o =$ Interior otherwise, proving Statement 2. □

IV. ANALYSIS

A. Computational Complexity

Function VORONOI-NEIGHBORS calls NEXT-NEIGHBOR $O(|V_i|)$ times, where $|V_i|$ is the cardinality of the Voronoi neighbor set V_i. The function NEXT-NEIGHBOR performs $O(N_i)$ operations, where N_i is the number of robot i's detection neighbors. Thus, VORONOI-NEIGHBORS is $O(|V_i|N_i)$. The average $|V_i|$ per robot is at most six [7]; effectively, $|V_i|$ is constant and VORONOI-NEIGHBORS is $O(N_i)$.

The distributed Voronoi algorithm of [8] is also $O(N_i)$.[3] Unlike VORONOI-NEIGHBORS, however, the algorithm of [8]

[3]It is $O(1)$ if the communication neighbors are already sorted by distance.

requires the robots to know their absolute positions. Thus, when only inter-robot distances are available, a localization method (e.g., [11], [12], [14], [15]) must first be performed. Solving this problem, however, generally dominates computation time because embedding the detection graph $\mathcal{G}$ in $\mathbb{R}^2$ is NP-hard [16]. By purposefully avoiding localizing some agents, localization can be solved in polynomial time with respect to the number of detection neighbors [15].

Rather than localizing and then finding Voronoi neighbors, VORONOI-NEIGHBORS can aid localization: for example, the algorithm of [15] is more efficient if it operates on the (relatively few) Voronoi neighbors rather than on all of the detection neighbors. Also, by modifying VORONOI-NEIGHBORS to track the sequence of the Voronoi neighbors along the Delaunay path, these neighbors can be localized in $O(|V_i|)$ time, with each robot using the Delaunay triangles to compute local coordinates. By transmitting these coordinates, in sequence, the robots can find the translation, rotation, and flip between neighboring coordinate systems. Although more efficient, noise may make this method less accurate than others because it does not use all available measurements.

B. Limited Detection

To identify all of its Voronoi neighbors, robot i must detect (communicate with and sense) them. That is, V_i must be a subset of robot i's detection neighbors. Additionally, two robots that are opposite a shared Delaunay edge (e.g., i and l for Delaunay triangles $\{i, j, k\}$ and $\{l, j, k\}$) must measure each other (but need not communicate). These communication and measurement requirements are more restrictive than those necessary for localization (see, e.g., [14]); however, they also provide a stronger guarantee. They ensure that every robot can localize its own Voronoi neighbors. For radius-limited detection graphs, where every robot detects every other robot within a radius R, it is sufficient for R to be at least twice the length of the longest Delaunay edge. With high enough robot density, this condition is satisfied [7].

C. Noise

Noisy measurements reduce the accuracy of all Voronoi neighbor identification algorithms. Noise can also cause asymmetry in the Voronoi neighbor relation: robot i may identify robot j as a Voronoi neighbor without robot j viewing robot i as a neighbor. To repair this inconsistency we introduce an extra step; every robot i sends V_i to its detection neighbors. When robot j receives V_i it checks if $j \in V_i$ and if so adds i to V_j. After this consistency step, $j \in V_i$ if and only if $i \in V_j$.

We now use the concept of distortion (introduced in Section II-D) to quantify the error in the algorithm's inputs (i.e., the measurements) and outputs (i.e., the computed Delaunay graph). Using distortion as a distance between graphs is well-established [13]. Let $\overline{\mathcal{G}}$ be the graph with the same topology as the detection graph $\mathcal{G}$, but with edge weights equal to the actual Euclidean distance d_{ij} between robots (i.e., an edge with weight d_{ij} in $\overline{\mathcal{G}}$ exists between robots i and j if they detect each other). We use the *detection distortion* δ_m between the metric spaces corresponding to $\overline{\mathcal{G}}$ and $\mathcal{G}$ to quantify the overall measurement error introduced by noise.[4]

To gain intuition, suppose the only edge in $\mathcal{G}$ with a possibly inaccurate measurement is between robots i and j. If edge $e_{ij} = d_{ij}$ then the measurement has no noise, $\mathcal{G} = \overline{\mathcal{G}}$, and $\delta_m = 1$ (note: $\delta_m \geq 1$ by definition). If $e_{ij} < d_{ij}$ then $\delta_m = \frac{d_{ij}}{e_{ij}}$ and if $e_{ij} > d_{ij}$ then $\delta_m = \frac{e_{ij}}{d_{ij}}$, so whether the sensed distance is longer or shorter than the actual distance, $\delta_m > 1$. Given fixed measurement error $e_{ij} - d_{ij}$, distortion increases with $|d_{ij}|$: this concept fits the idea that, for example, a $1\,\mathrm{m}$ error is worse for an actual distance of $10\,\mathrm{m}$ than for $100\,\mathrm{m}$.

The algorithm, when operating on noisy measurements, determines the computed graph $\mathcal{G}_v$. An edge with weight e_{ij} from $\mathcal{G}$ exists between robots i and j in $\mathcal{G}_v$ if they identify each other as Voronoi neighbors. The *Delaunay distortion* δ_v is the distortion between the metric spaces corresponding to the actual Delaunay graph $\overline{\mathcal{G}}_v$ and $\mathcal{G}_v$; it measures the algorithm's error. Suppose, for example, that robots i and j mistakenly identify each other as Voronoi neighbors, but all other identifications are correct. Then between robots i and j an edge exists in $\mathcal{G}_v$ that does not exist in $\overline{\mathcal{G}}_v$. Generally, this additional edge decreases the distance between robots i and j in $\mathcal{G}_v$ relative to their distance in $\overline{\mathcal{G}}_v$; the farther apart robots i and j are (based on the number of edges on the shortest path between them), the larger this decrease and the larger the distortion. Thus, distortion matches the intuition that misidentifying robots close to Voronoi neighbors results in less error than misidentifying distant robots. In Section V-B, we analyze our results using distortions δ_m and δ_v.

V. Experiment and Simulations

We use an experimentally validated sensor model to test our algorithm in simulation. The simulations allow testing many more scenarios than with experiments alone. The experiment provides confidence that the simulation accurately captures sensor noise, which is the main difference between the theoretical and practical aspects of our algorithm.

A. Experiment

We placed two XBee radios at distances from $8.89\,\mathrm{cm}$ to $74.294\,\mathrm{cm}$ apart at intervals of $0.64\,\mathrm{cm}$. We sent approximately 50 packets at each location and recorded the received signal strength indicator (RSSI). These data are plotted in Figure 7. This signal is highly quantized, providing only 69 distinct values [17]. Over distances between $8.89\,\mathrm{cm}$ to $33.04\,\mathrm{cm}$, however, the data match the physical expectation that signal power drops proportionally to the inverse square of distance. Thus, over these distances we fit a quadratic curve mapping RSSI signals to distances (see Figure 7).

We use the experimental data to generate simulated sensor data. Given an actual inter-robot distance d, the model finds the distances d_i and d_{i+1} (locations where experimental data were gathered, so $d_{i+1} - d_i = 0.64\,\mathrm{cm}$) such that $d_i < d < d_{i+1}$ We then randomly draw an RSSI value from the data associated with distances d_i and d_{i+1}, weighted according to

[4]Disconnected graphs do not correspond to metric spaces with finite metrics. We disregard the disconnected case because when the agent density is high enough both the detection and computed graphs are connected.

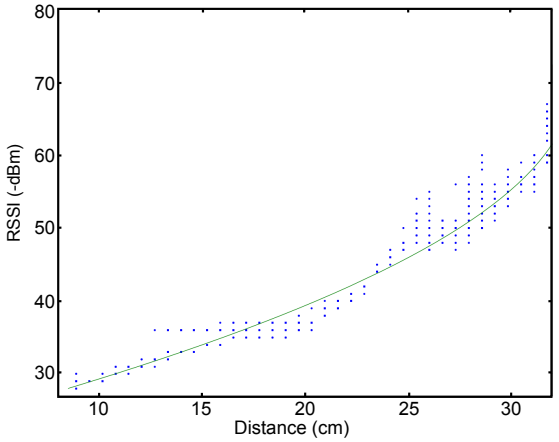

Fig. 7. RSSI signal (-dBm) vs. Distance (cm). Higher values indicate a weaker signal. The green curve is the quadratic fit with $y = -0.0141x^2 + 1.9614x - 35.365$. Each plotted point may represent several readings of the same RSSI value at a given distance.

the distribution of the actual data at these locations and the distance from d to d_i and to d_{i+1}.

B. Simulations

We ran simulations with robots uniformly randomly placed inside a $254\,\mathrm{cm}$ by $254\,\mathrm{cm}$ domain. The robots sense distances using simulated XBee RSSI data, communicate measurements to their neighbors, run the VORONOI-NEIGHBORS algorithm, and perform the consistency update. Robots more than $33\,\mathrm{cm}$ apart cannot sense or communicate with each other. We tested the algorithm for $N = 200 \ldots 1000$ robots with 100 trials for each N (a total of 80,000 trials). We disregard trials with disconnected detection or computed graphs: less than 2% of trials with more than 300 robots had a disconnected Delaunay graph, and most of these had a disconnected detection graph.

Figure 8 plots the Delaunay distortion δ_v versus the detection distortion δ_m. It indicates that the error is proportional to the inaccuracy of the XBee distance measurements. The ratio $\frac{\delta_v}{\delta_m}$ is bi-modal, clustered around 1 and 2, with $\frac{\delta_v}{\delta_m} < 6.5$. When $\frac{\delta_v}{\delta_m} \approx 1$ most output error is from the noisy edge weights of $\mathcal{G}_v$ rather than incorrect identifications; therefore, the topology of $\mathcal{G}_v$ and $\overline{\mathcal{G}}_v$ is similar and most identifications are correct. Larger $\frac{\delta_v}{\delta_m}$ indicates more error: the clustering around 2 indicates that the non-locality of misidentified Voronoi neighbors is limited by the measurement error: robots that are far away relative to the magnitude of measurement inaccuracy generally do not misidentify each other. Thus our algorithm provides reasonable estimates of the Voronoi neighbors and degrades smoothly with respect to errors due to XBee RSSI measurement noise.

VI. Conclusion

Our algorithm allows robots to determine their Voronoi neighbors using only inter-robot distances, without placing those neighbors in a local coordinate system. The method results in a more efficient determination of the Voronoi neighbors, compared to localizing these neighbors first, and also can aid in more efficient localization. We demonstrate the practicality of the algorithm using an experimentally validated distance sensor model based on XBee RSSI data.

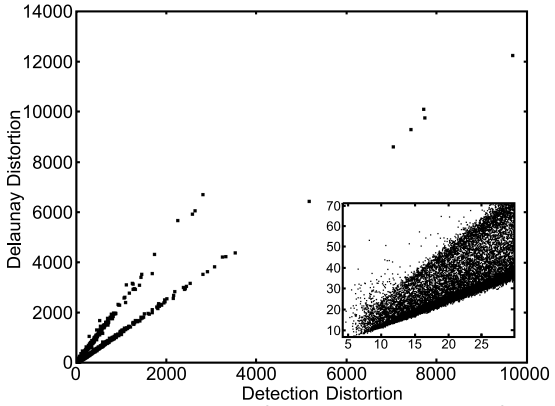

Fig. 8. **Main:** Delaunay distortion δ_v vs. detection distortion δ_m. Note that most trials were in the lower left corner (see inset); particularly erroneous measurements led to more detection distortion resulting in proportionally more Delaunay distortion. **Inset:** A closer view: most trials cluster around $\frac{\delta_v}{\delta_m} = 1$.

References

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

[2] M. Schwager, D. Rus, and J.-J. Slotine, "Decentralized, adaptive coverage control for networked robots," *The International Journal of Robotics Research*, vol. 28, no. 3, pp. 357–375, 2009.

[3] M. Schwager, J. Mclurkin, and D. Rus, "Distributed coverage control with sensory feedback for networked robots," in *in Proceedings of Robotics: Science and Systems*, 2006.

[4] S. Martínez, "Distributed interpolation schemes for field estimation by mobile sensor networks," *Control Systems Technology, IEEE Transactions on*, vol. 18, no. 2, pp. 491–500, March 2010.

[5] M. L. Elwin, R. A. Freeman, and K. M. Lynch, "Environmental estimation with distributed finite element agents," in *55th IEEE Conference on Decision and Control*, Dec 2016, pp. 5918–5924.

[6] M. Kolahdouzan and C. Shahabi, "Voronoi-based k nearest neighbor search for spatial network databases," in *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, ser. VLDB '04. VLDB Endowment, 2004, pp. 840–851.

[7] A. Okabe, B. Boots, K. Sugihara, S. N. Chiu, and D. G. Kendall, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2nd ed. John Wiley and Sons, 2000.

[8] M. Cao and C. Hadjicostis, "Distributed algorithms for Voronoi diagrams and application in ad-hoc networks," UIUC Coordinated Science Laboratory, Tech. Rep. UILU-ENG-03-2222,DC-210, 2003.

[9] B. A. Bash and P. J. Desnoyers, "Exact distributed Voronoi cell computation in sensor networks," in *2007 6th International Symposium on Information Processing in Sensor Networks*, April 2007, p. 236.

[10] W. Alsalih, K. Islam, Y. Núñez Rodríguez, and H. Xiao, "Distributed Voronoi diagram computation in wireless sensor networks," in *Proceedings of the Twentieth Annual Symposium on Parallelism in Algorithms and Architectures*, ser. SPAA '08. New York, NY, USA: ACM, 2008, pp. 364–364.

[11] U. A. Khan, S. Kar, and J. M. F. Moura, "Linear theory for self-localization: Convexity, barycentric coordinates, and Cayley–Menger determinants," *IEEE Access*, vol. 3, pp. 1326–1339, 2015.

[12] G. Mao, B. Fidan, and B. D. O. Anderson, "Wireless sensor network localization techniques," *Computer Networks*, vol. 51, no. 10, pp. 2529–2553, July 2007.

[13] J. Matoušek, *Lectures on discrete geometry*. Springer New York, 2002, vol. 108.

[14] T. Eren, O. K. Goldenberg, W. Whiteley, Y. R. Yang, A. S. Morse, B. D. O. Anderson, and P. N. Belhumeur, "Rigidity, computation, and randomization in network localization," in *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 4, March 2004, pp. 2673–2684 vol.4.

[15] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust distributed network localization with noisy range measurements," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*. New York, NY, USA: ACM, 2004, pp. 50–61.

[16] J. B. Saxe, "Embeddability of weighted graphs in k-space is strongly NP-hard," in *Proceedings of the 17th Allerton Conference on Communication, Control, and Computing*, 1979, pp. 166–179.

[17] *XBee / XBee-PRO RF Modules*, v1.xEx ed., Digi International, 2009.