2012 Third International Conference on Emerging Applications of Information Technology (EAIT)

A Distributed Algorithm for Computation of Exact Voronoi Cell in a Multi-Robotic System*

K.R. Guruprasad
Department of Mechanical Engineering
National Institute of Technology Karnataka, Surathkal, India.
Email: krgprao@gmail.com

Prithviraj Dasgupta
Department of Computer Science
University of Nebraska, Omaha, NE, USA.
Email: pdasgupta@mail.unomaha.edu

Abstract—**In this paper we propose an algorithm for distributed computation of Voronoi cell in a multi-robotic system. Each of the robots is assumed to know its own position and position of all other robots. The robots compute their Voronoi cells based only on this positional information, without any additional communication and cooperation with other robots.**

I. Introduction

Voronoi partitioning has been used as a partitioning technique for multi-robot area coverage and sensor coverage in several multi-robotic systems (MRS) [1] and sensor networks [2], [3]. The Voronoi partition of a space is calculated using the positions of robots within it and each robot operates within, and, consequently, needs to be aware of only its Voronoi cell and not the entire Voronoi partition. However, in most of the applications involving Voronoi partitioning techniques mentioned above, each robot computes the entire Voronoi partition, extracts its Voronoi cell and discards the information about the remaining cells. The discarded information corresponds to considerable amounts of useless computation done by each robot, and incurs unnecessary expenditure of energy and time. In this paper, we attempt to address this deficit by developing a distributed Voronoi partitioning technique where each robot computes only its own Voronoi cell. Unlike most existing techniques for distributed Voronoi partitioning, we follow a more structured approach based on relative robot positions in polar coordinate system.

II. Distributed Voronoi cell computation

Consider N robots in a multi-robot system (MRS). Let $\mathcal{P} = \{p_1, p_2, \ldots, p_N\}$ be the configuration of the MRS, where $p_i \in \mathbb{R}^2$ is the position of the i-th robot. Let $I_N = \{1, 2, \ldots, N\}$ be an index set. By a slight abuse of notation, we use p_i to refer to both the i-th robot and its position in the space. All the robots know the configuration $\mathcal{P}$. This can be achieved by a broadcast communication amongst the robots. Let $\mathcal{V} = \{V_i | i \in I_N\}$ be the Voronoi partition generated by $\mathcal{P}$ as a node set, with

$$V_i = \{q | \|p_i - q\| \leq \|p_j - q\|, \forall j \in I_N\}$$

*This work has been supported as part of the COMRADES project supported by the U.S. DoD Office of Naval Research, grant no. N0000140911174.

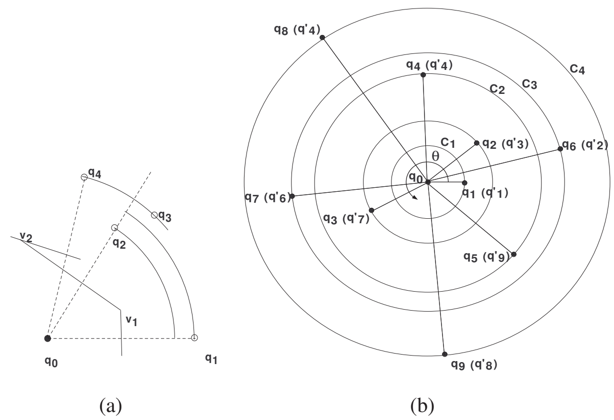


Fig. 1. a) Motivation for using relative configuration in polar coordinate system. b) he robot p_i re-indexes all robots based on the relative position in a polar coordinate system with $p_i = q_0$ as center.

Nodes i and j are considered Voronoi neighbors (or neighbors in the Delaunay graph $\mathcal{G}_D$), if the corresponding Voronoi cells V_i and V_j share a common edge.

Problem statement: For each $i \in I_N$, given $\mathcal{P}$, the i-th robot should compute the corresponding Voronoi cell V_i, and its Voronoi neighbors.

A. Preprocessing

The Voronoi cell V_i of the robot located at p_i, and its Voronoi neighbors, depend on the positions and orientations of the remaining robots. While calculating the Voronoi cell of each robot, selecting the set of its Voronoi neighbors based on Euclidean distances with positions of robots represented in Cartesian coordinates might lead to complicated analysis and calculations. For example, in Figure 1(a), robots q_1 and q_2 are the Voronoi neighbors for robot q_0. Robots q_3 and q_4 are equidistant from q_0 but it is easy to see that q_3 is not q_0's Voronoi neighbor, while q_4 could be a Voronoi neighbor, depending on the relative position and orientation of other robots. In contrast, representing relative robot positions using a polar coordinate system provides a more succinct way to enable the computation of Voronoi cells.

The i-th robot, p_i, constructs two ordered sets of robots to represent the configuration of its neighboring robots in the polar coordinate space with itself as the origin. The first set

978-1-4673-1827-3/12/$31.00 ©2012 IEEE

${}^iQ = \{{}^iq_1, {}^iq_2, \dots, {}^iq_{N-1}\}$ contains the neighboring robots of p_i sorted in order of increasing distance (radius) from p_i. Ties in radii are broken by ordering the robots equidistant from p_i in increasing order of angles with the line joining p_i and ${}^iq_1$, the closest robot to p_i. If more than one robot are on C_1, then one of these robots[1] is randomly chosen as q_1. The second set ${}^iQ' = \{{}^iq'_1, {}^1q'_2, \dots, {}^iq'_{N-1}\}$ contains the neighboring robots of p_i sorted in order of increasing angle with the line joining p_i and ${}^iq_1$, the closest robot to p_i as the base angle. Ties on angle are broken by ordering robots in increasing distance (radius). For the sake of legibility, we rename robot p_i as q_0. An example illustrating the construction of the sets ${}^iQ, {}^iQ'$ is illustrated in Figure 1(b). With $q_0(= p_i)$ as center, let ${}^iC_1, {}^iC_2, \dots {}^iC_K$, $K \leq N-1$, denote virtual circles of increasing radii passing through one or more robots in ${}^iQ$; circle ${}^iC_1$ passes through ${}^iq_1$ and so on. Let ${}^ir_k$ be the radius of circle ${}^iC_k$. In the following, when the context is unambiguous, we drop the superscript i to simplify the notation and refer to ${}^iq_j$ as q_j, ${}^iC_k$ as C_k, and ${}^ir_k$ as r_k. For brevity, we use q_k (or q'_k) to refer to both the robot itself and its position.

After computing the relative configuration of its neighboring robots, the i-th robot computes its Voronoi cell in two distinct phases - the *expansion* phase and the *contraction* phase, which are described in the following sections.

B. Expansion Phase

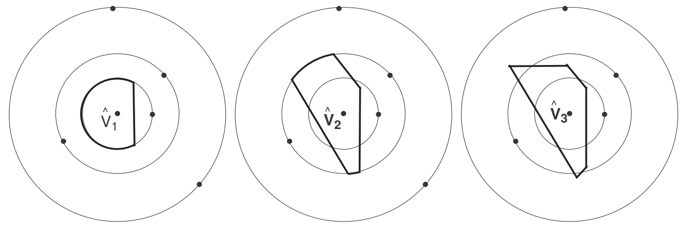
Assume that the i-th robot has access only to $\bar{B}(p_i, R_i)$, the closed disc of radius R_i centered at p_i. If there are no robots within a distance of R_i, then the constrained Voronoi cell is $\bar{B}(p_i, R_i)$ itself. We call R_i as pseudo sensor range, as this limit on the range is only used for the purpose of computation, and not the real limitation of the robot. The i-th robot computes the Voronoi cell starting with $R_i(1) = r_1$, and expands the constrained Voronoi cell incrementally by setting $R_i(n) = r_n$ at n-th step. The procedure is discussed formally in the following.

Let N_k be the number of robots on $C_k, k \in I_K$. Note that in non-degenerate conditions, $N_k \leq 3, \forall k \in I_K$. Let $\mathcal{N}_k = N_1 + N_2 + \dots + N_k$, the number of robots on or inside C_k. Let $H(q_0, p)$ be the half plane defined by perpendicular bisector of $q_0 \leftrightarrow p$ containing q_0, for any $p \in \mathcal{P} \setminus \{p_i\}$, and $D_k = \bar{B}(p_i, r_k)$.

The robot starts with $R_i(1) = r_1$ and finds the Voronoi cell at the n-th step as,

$$\hat{V}_n = \{D_n \cap \{\bigcup_{l=1}^{\mathcal{N}_n} H(q_0, q_l)\}, n \in I_K \tag{1}$$

If we denote $V_i(\mathcal{P}_n)$ as the Voronoi cell corresponding to p_i with $\mathcal{P}_n = \{p_j | \quad \|p_i - p_j\| \leq r_n, j \in I_{N-1}\} \subset \mathcal{P}$, the set of robots/nodes on or within the virtual circle C_n as node set, then $\hat{V}_n = D_n \cap V_i(\mathcal{P}_n)$ gives the portion of Voronoi cell within D_n. Note the boundary of V_n is made up of line segments corresponding to perpendicular bisectors and arcs on



(a) (b) (c)

Fig. 2. Computation of approximate Voronoi cell in the expansion phase. The candidate Voronoi cell at each step is shown by the region enclosed by bold lines.

the virtual circle C_n. If p_i is in the interior of the convex hull of $\mathcal{P}$, the expansion phase continues until either the Voronoi cell $\hat{V}_n$ is a bounded polygon within D_n, or until the largest virtual circle C_K centered at p_i is reached ($n = K$) without finding a bounded polygon. A special case occurs if p_i lies on the boundary of the convex hull of $\mathcal{P}$, and p_i can never have a polygonal Voronoi cell as the Voronoi cell is unbounded. In this case, the expansion phase terminates when there are no arcs in the portion of the boundary of $\hat{V}_n$ within the convex hull of $\mathcal{P}$. Figure 2 illustrates the expansion phase for a robot p_i with the same configuration of neighboring robots shown in Figure 1(b). The expansion phase terminates at the 3rd circle as soon as a bounded polygon is found. In the following section, we describe a distributed algorithm that can be used by each robot for constructing its candidate Voronoi cell using this expansion technique.

The construction of the candidate Voronoi cell during the expansion phase proceeds into two steps as shown in Algorithm 1. In the first step, we construct a chord L_j^k [2] corresponding to each neighboring robot q'_j within C_k. For this, we first look at the intersection of b_{0j}, the perpendicular bisector of line joining q_0 and q'_j (j-th node on Q', the relative configuration sorted based on θ), with the circle C_k. Let the points of intersection of b_{0j} with C_k be (r_k, θ_j^s) and (r_k, θ_j^e). We already know r_k, and θ_j^s and θ_j^e can be calculated as:

$$\begin{aligned} \theta_j^s &= \mod{}_{2\pi}(\theta_j + \delta\theta_j) \\ \theta_j^e &= \mod{}_{2\pi}(\theta_j - \delta\theta_j) \end{aligned} \tag{2}$$

where, $\delta\theta_j = cos^{-1}\left(\frac{r_j}{2r_k}\right)$. Note that it is easy to see that $2\pi/3 \leq \delta\theta_j < \pi/2$. We store the set of chords corresponding to each q'_j within C_k in an ordered set called $\mathcal{L}$.

In the second step, we find the intersection points of the chords constructed in $\mathcal{L}$ from the first step and check to see if they form a bounded polygon within circle C_k. If such a bounded polygon is found, we terminate with the candidate Voronoi cell, otherwise we continue to the next larger circle C_{k+1}.

The checking of chords to find intersections requires some insight. To enable our calculations, we construct an ordered set of polar angles of the chords, represented with their start and end points in polar coordinates, in the set Θ, in

[1]It is easy to show that all robots on C_1 are Voronoi neighbors of p_i.

[2]For brevity, we drop the superscript k when the context is clear.

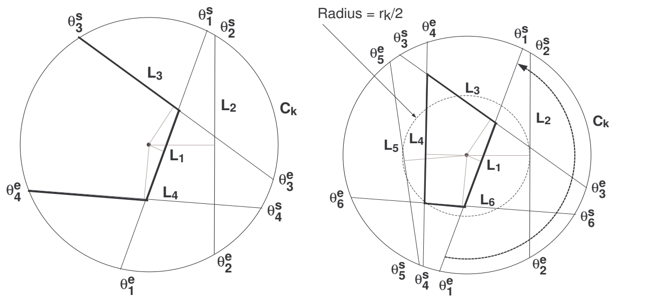
expansion(K, Q')
Input: K, Q'; //K: no. of virtual circles in neighbor configuration of robot p_i; Q': angle ordered set of neighboring robots of p_i in polar coords
Output: $\hat{V}, \mathcal{L}$; //$\hat{V}$: ordered vertex set representing candidate Voronoi cell of robot p_i if successful, null set if unsuccessful; $\mathcal{L}$:angle-sorted list of chords in circle C_k
$\hat{V} \leftarrow \emptyset$;
$q_0 \leftarrow$ origin of polar coord system corresponding to location of p_i;
for $k = 2$ *to* K **do**
  $\mathcal{L} \leftarrow \emptyset; \Theta \leftarrow \emptyset$;
  for $j = 1$ *to* k **do**
    $L_j \leftarrow$ Perp. bisector of line $(q_0, q_j') : q_j' \in Q'$;
    $(\theta_j^s, \theta_j^e) \leftarrow$ Angles of extremities of chord L_j with q_0 within circle C_k of radius r_k;
    $\mathcal{L} \leftarrow \mathcal{L} \cup L_j$;
    $\Theta \leftarrow \Theta \cup \{\theta_j^s, \theta_j^e\}$
  end
  // $\mathcal{L}$ is already in ascending order of index j;
  Sort Θ in ascending order;
  // remove redundant chords from $\mathcal{L}$ using checkChords method
  $boundedPoly, convexBounds \leftarrow checkChords(\Theta, \mathcal{L}, Q')$;
  if $boundedPoly = true$ **then**
    $\hat{V} \leftarrow$ constructPoly($\mathcal{L}, convexBounds$);
    break;
  end
end
return $\hat{V}, \mathcal{L}$;

Algorithm 1: Algorithm used by robot p_i during the expansion phase of the distributed Voronoi cell computation

checkChords($\Theta, \mathcal{L}, Q'$)
Input: $\Theta, \mathcal{L}, Q'$; // Θ: sorted list of endpoints of chords in circle C_k, $\mathcal{L}$: angle-sorted list of chords in circle C_k, Q': angle ordered set of neighboring robots of $p_i (= q_0)$ in polar coords
Output: $boundedPoly, convexBounds$; //$boundedPoly$: boolean values indicating if chords in $\mathcal{L}$ form a close polygon, $convexBounds$: ordered pair of chords in $\mathcal{L}$ if p_i is on a convex hull with its neighboring robots
foreach $L_j \in \mathcal{L}$ **do**
  $intersect[L_j] \leftarrow \emptyset$;
end
foreach $(\theta_j^s, \theta_j^e) \in \Theta$ **do**
  $\Theta_{diff,j} \leftarrow$ ordered set of endpoints of chords lying between (θ_j^e, θ_j^s) in Θ;
  if $\exists m$ *s.t.* $\theta_m^s \wedge \theta_m^e \in \Theta_{diff,j}$ **then**
    Remove L_m from $\mathcal{L}$;
  end
  else if $\exists m$ *s.t. either* θ_m^s *or* $\theta_m^e \in \Theta_{diff,j}$ **then**
    $intersect[L_j] \leftarrow intersect[L_j] \cup L_m$;
  end
end
//check to see if remaining chords in $\mathcal{L}$ form a closed polygon $boundedPoly \leftarrow false$;
$convexBounds \leftarrow \emptyset$;
foreach $L_j \in \mathcal{L}$ **do**
  $L_m \leftarrow L_j.next()$;
  // next() gives the next chord in $\mathcal{L}$ with wrap around
  if $L_m \in intersect[L_j]$ **then**
    boundedPoly $\leftarrow$ true;
  end
  else
    //check if polygon will be unbounded because p_i is on convexHull of neighboring robots
    $\{\theta_{min}, \theta_{max}\} \leftarrow checkConvexHull(Q')$;
    // get the angle of radial lines bisecting chords L_j and L_k resp.
    $\theta_j \leftarrow (\theta_j^s + \theta_j^e)/2$; $\theta_m \leftarrow (\theta_m^s + \theta_m^e)/2$;
    if $\{\theta_j, \theta_m\} = \{\theta_{min}, \theta_{max}\}$ **then**
      $convexBounds \leftarrow L_j, L_m$;
      $boundedPoly \leftarrow true$;
    end
  end
  else
    $boundedPoly \leftarrow false$;
    break;
  end
end
return $boundedPoly, convexBounds$;

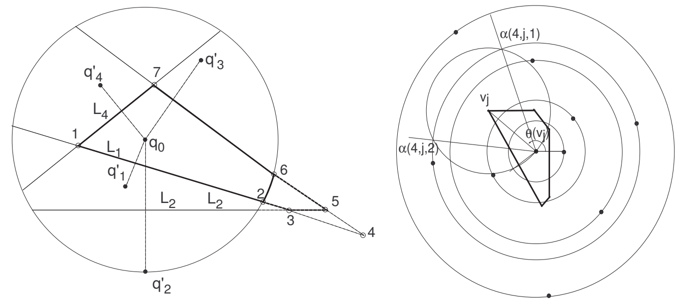
Algorithm 2: Method used within expansion phase to remove non-intersecting lines in $\mathcal{L}$

the method *expansion* given in Algorithm 1. Then, in the *checkChords* method given in Algorithm 2, for each chord L_j, we extract the set of chords that have either their start or end point, or both, between L_j's ending and starting points (in that order), in an ordered set $\Theta_{diff,j}$, starting with the end point of the first chord L_1 in $\mathcal{L}$. The chords which have only one start or end point between L_j's end and start points intersect L_j, while those that have both end points do not intersect L_j. We store the intersects of L_j with other chords in a set called $intersect$ that is indexed by L_j, while we remove chords that do not intersect L_j from the set of chords $\mathcal{L}$. As an example, consider the set Θ from our running example shown in Figure 1(b) and re-illustrated in Figure 3(b), and $\Theta = (\theta_1^e \ \theta_2^e \ \theta_6^s \ \theta_3^e \ \theta_2^s \ \theta_1^s \ \theta_4^e \ \theta_3^s \ \theta_5^e \ \theta_6^e \ \theta_5^s \ \theta_4^s)$. Here $\Theta_{diff,1} = \{\theta_2^e \ \theta_6^s \ \theta_3^e \ \theta_2^s\}$. Since both the start and end points of L_2, θ_2^e and θ_1^e appear in $\Theta_{diff,1}$, we remove L_2 from $\mathcal{L}$, and the entries corresponding to L_2, *viz.*, θ_2^e and θ_2^s, from Θ. Looking at Figure 3(b), this makes sense because chords L_2 and L_1 do not intersect with each other. The remaining entries in $\Theta_{diff,1}$ are θ_6^s and θ_3^e, corresponding to chords L_6 and L_3 respectively, that has either one start point or one end point in $\Theta_{diff,1}$. And so, $intersect[L_1] = \{L_3, L_6\}$. Proceeding in this manner, we



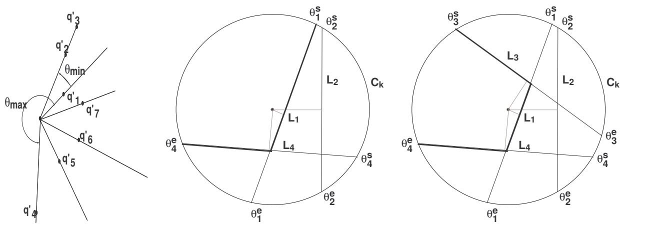
(a) (b)

Fig. 3. Expansion phase is (a) not terminated, and (b) terminated, when the i-th robot is in the interior of $Co(\mathcal{P})$.



(a) (b)

Fig. 5. a) The constrained Voronoi cell $\hat{V}_K$ is $1 \rightarrow 2 \rightarrow 6 \rightarrow 7$, is not a polygon. After termination of expansion phase, $\hat{V}_K$ is extended into a polygon $1 \rightarrow 4 \rightarrow 7$, while actual Voronoi cell V_i is $1 \rightarrow 3 \rightarrow 5 \rightarrow 7$. b) Illustration of contraction phase.



(a) (b) (c)

Fig. 4. a) Illustration of a node on convex hull. Nodes q'_2 and q'_4 are neighbors of q_0 in $\mathcal{G}_D$. Expansion phase is b) not terminated, andc) terminated, when the i-th robot is on the boundary of $Co(\mathcal{P})$, where ConvexBounds $= (L_3, L_4)$.

get, $intersect[L_3] = \{L_1, L_4\}$, $intersect[L_4] = \{L_4, L_6\}$ (chord L_5 gets removed while inspecting $\Theta_{diff,4}$), and $intersect[L_6] = \{L_4, L_1\}$.

checkConvexHull(Q')
Input: Q' // set of neighboring robots of p_i order in increasing polar angles w.r.t. p_i
if $\exists\, q_j \in Q', j \in I_{N-2} s.t. (\theta(q_{j+1}) - \theta(q_j)) > \pi$ **then**
  $(\theta_{min}, \theta_{max}) \leftarrow (\theta(q_j), \theta(q_{j+1}))$;
  return true;
end
else if $\theta(q_{N-1}) \leq \pi$ **then**
  $(\theta_{min}, \theta_{max}) \leftarrow (\theta(q_{N-1}), 0)$;
  return true;
end
else
  return false;
end

Algorithm 3: Algorithm for checking if p_i is on a convex hull w.r.t its neighboring robots

Checking if the final set of chords forms a bounded polygon is fairly intuitive - we check if every chord intersects with its successive chord in the ordered set. If this condition is not satisfied for one successive pair of chords, we cannot find a bounded polygon within the current circle. For example, in Figure 3(b) chords L_3 and L_4 do not intersect and the set of chords form an unbounded polygon. We then continue to the calculations of chords for the next larger circle. Continuing the example from Figure 3(a), we see that in Figure 3(a) the final set of chords L_1, L_3, L_4 and L_6 intersect with each other and form a bounded polygon. Finally, if we reach the largest circle for the robot and are still unable to find a bounded polygon within D_K, the expansion phase is terminated and a polygon is constructed based on existing lines in the chord set $\mathcal{L}$. Figure 5(a) gives an illustration of such a scenario.

A special case of an unbounded polygon occurs when the robot p_i calculating the Voronoi cell lies on the convex hull of the set of robots $\mathcal{P}$. This condition is illustrated in Figure 4(a) and checked in the latter part of the *checkChords* method in Algorithm 2 using the checkConvexHull method shown in Algorithm 3. For each pair of chords in the final set of chords $\mathcal{L}$, we check if their respective perpendicular bisectors correspond to a pair of successive edges on the convex hull of the set of robots $\mathcal{P}$. Formally, the conditions that test if $p_i \in \partial(\text{Co}(\mathcal{P}))$ are given in the following lemmas.

Lemma 1: A point $p_i \in \mathcal{P}$ is on $\partial(\text{Co}(\mathcal{P})$, if and only if, there exists $j \in I_{N-2}$, such that, $\theta(q'_{j+1}) - \theta(q'_j) > \pi$, or, $\theta(p'_{N-1}) < \pi$.
The proof is fairly straight forward and is skipped here.

Lemma 2: For a robot $p_i \in \partial(\text{Co}(\mathcal{P}))$, closest nodes on radial lines along $\theta_{\max}$ and $\theta_{\min}$ directions are neighbors in $\mathcal{G}_D$.

The result is fairly intuitive and we skip the formal proof. The expansion phase is terminated even when $\hat{V}_n$ is not a polygon as illustrated in Figure 4(c). Here, the chords L_3 and L_4 correspond to nodes on radial lines corresponding to $\theta_{\min}$ and $\theta_{\max}$, respectively. Whereas, as illustrated in Figure 4(b), when $L_3 \notin \mathcal{L}$, the expansion phase is not terminated.

The *constructPoly* method returns a set of points corresponding to the intersections of the chords returned by *checkChords* method if a bounded polygon can be found. Otherwise, if the convex hull case occurs, it returns the appropriate edges from the convexHull along with the intersection points of the intersecting chords.

C. Contraction Phase

In the second phase called *contraction phase*, the Voronoi cell is contracted by identifying candidate nodes or neighbors.

Contraction($\hat{V}, \mathcal{L}$)
Input: $\mathcal{L}, convexBounds$; //$\mathcal{L}$: minimal list of chords (angle-sorted) returned by constructPoly method, V: ordered vertex set representing candidate Voronoi cell of robot p_i returned by constructPoly method
Output: V;// ordered set of vertices representing the Voronoi cell, $\hat{N}'$ set of indexes of robots in Q' which are Voronoi neighbors of robot p_i.

```
N̂′ ← {j | L_j ∈ L};
j ← 1;
d_j ← ‖q_o − v_j‖
// v_j is the current vertex of V being checked for
possible contraction of V.
while j ≤ | V | do
    m = max_k {k | r_k ≤ 2d_j}
    l ← 1;
    //Start with first circle.
    while (l ≤ m) do
        Q′_l ← {q′_k | r(q′_k) = r_l};
        // all robots on circle C_l.
        α_1 ← θ(v_j) − cos^{-1}(r_l/2d_j);
        α_1 ← θ(v_j) − cos^{-1}(r_l/2d_j);
        //θ(v_j) is the angle between lines joining q_0 to v_j
        and that joining q_0 and q_1.
        if ∃q′_k ∈ Q′_l s.t. (α_1 ≤ θ(q′_k) ≤ α_2) ∧ (k ∉ N̂′)
        then
            v_1 ← findIntersection(L_{j1}, L_k);
            v_2 ← findIntersection(L_k, L_{j2});
            // the vertex v_j is formed by lines
            L_{j1}, L_{j2} ∈ L with L_{j2} = L_{j1}.next.
            V ← V \ {v_j};
            V ← V ∪ {v_1, v_2};
            // new vertexes v_1 and v_2 are inserted in
            order, after v_{j−1}.
            N̂′ ← N̂′ ∪ {k};
            l = m + 1; // Do not check any more robots
            for v_j.
            j ← j − 1;
        end
    end
    j ← j + 1;
end
return V, N̂′
```

Algorithm 4: Algorithm used to contract the candidate Voronoi cell during the contraction phase

This is achieved by checking for candidate nodes which contract the Voronoi cell, for each vertex of the candidate polygon, using following condition [4], [5].

Lemma 3: Consider a bounded Voronoi cell $\hat{V}_k$ corresponding to a node p. A node q reduces $\hat{V}_k$ iff there exists a vertex v of $\hat{V}_k$ such that $\|q - v\| < \|p - c\|$.

Instead of checking all nodes, only selected nodes will be checked using the relative configuration information, thus reducing the number of checks. (In other words, instead of searching whole of $\mathcal{P} \setminus \{p_i\}$, only a subset of it is searched.)

Let $\alpha(l, j, 1)$ and $\alpha(l, j, 2)$ be angles corresponding to intersections of C_l and a circle $C(v_j, d_j)$, centered at v_j, the j-th vertex of current Voronoi polygon, and radius $d_j = \|v_j - q_0\|$. See Figure 5(b) for illustration.

$$\alpha(l, j, 1(2)) = \theta(v_j) - (+) \cos^{-1}(r_k/2d_j) \quad (3)$$

where $\theta(v_j)$ is the argument of point the v_j. For any vertex v_j, if there exists a node inside the disc $\bar{B}(v_j, d_j)$, then this node is a Voronoi neighbor of q_0 and will reduce the Voronoi cell. The corresponding perpendicular bisector is found and the Voronoi cell is updated. Now, if the expansion phase had terminated at C_n, then, for a vertex v_j, we need to check if there are nodes (robots) on circles C_l, within the angle range $(\alpha(l, j, 1), \alpha(l, j, 2))$, $m \geq l > n$, where $m = \arg\max_{j \in \{n+1, \ldots, K\}} \{r_j | r_j \leq d_j\}$, if $n < K$. If the expansion phase ended in last circle, then check if there exist any robots within $(\alpha(K, j, 1), \alpha(K, j, 2))$, on C_K. Note that number of vertexes of $\hat{V}_n$ is at most equal to number of neighbors of p_i in $\mathcal{G}_D$.

The contraction phase terminates when for every vertex v_j of the polygon $\hat{V}$, the approximate Voronoi cell, corresponding $\bar{B}(v_j, d_j)$ does not contain any node (robot).

III. Analysis of the Algorithm

In this section, we provide correctness results for the proposed distributed algorithm for computation of Voronoi cell.

Lemma 4: The expansion phase terminates in finite time.

This result is trivial as in worst case, the expansion phase terminates in K-th step and $K \leq N - 1$. □

We state a property of Voronoi partition here which will be used in the following to prove the correctness of the expansion phase.

Property 1. For any sets of nodes $\mathcal{P}_1, \mathcal{P}_2$ with $\mathcal{P}_1 \subseteq \mathcal{P}_2$, $\forall i : p_i \in \mathcal{P}_1, V_i(\mathcal{P}_2) \subseteq V_i(\mathcal{P}_1)$.

The Voronoi partition induces an undirected graph known as Delaunay graph, $\mathcal{G}_D$, where two nodes $i, j \in I_N$ are neighbors if the intersection of corresponding Voronoi cells V_i and V_j is a line segment. Set of neighbors of i is denoted as $\mathcal{N}_{\mathcal{G}_D}(i)$.

Lemma 5: If expansion phase terminates at n-th circle, $n < K$, then $V_i \subseteq \hat{V}_n$.

Proof. Let $Q_n = \{q \in \mathcal{Q} \mid r(q) \leq r_n\} \subset \mathcal{Q}$. Let $\mathcal{N}(\hat{V}_n)$ be such that if $q_j \in \mathcal{N}_n(\hat{V}_n)$, then $\hat{V}_n$ has an edge which is a line segment in b_{0j}. In other words, if $\hat{V}_n$ is the Voronoi cell of q_0, then $\mathcal{N}(\hat{V}_n)$ is the set of its Voronoi neighbors. Clearly, $\mathcal{N}_n(\hat{V}_n) \subset (\mathcal{Q} \setminus \{q_0\}) \subset \mathcal{P}$. Thus, by property 1, $V_i(\mathcal{P}) \subseteq \hat{V}_n$. □ computed.

Lemma 6: The contraction phase terminates.

Proof. At any stage in the contraction phase the candidate Voronoi cell has only a finite number of vertexes and for each vertex V_i, only a finite number of nodes can lie within $C(v_i, d_i)$. □

Theorem 1: The polygon or region at the end of contraction phase represents the Voronoi cell.

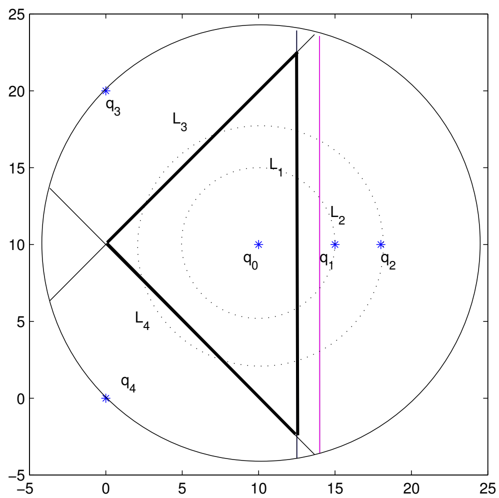


Fig. 6. Sample result obtained by implementation of proposed distributed Voronoi cell computation algorithm. The robot locations are marked with '*'.

Proof. Let $v(i)$ be the set of vertexes of $\hat{V}_n$, V' be the region obtained at the end of the contraction phase. By Lemma 5, $V_i \subseteq \hat{V}_n$. Further by Lemma 3, only nodes which can reduce $\hat{V}_n$ are the nodes within $B(v_j, d_j)$ for all $v_j \in v(i)$, which is checked in contraction phase. Thus by Lemma 3, $V' = V_i$. □

IV. ILLUSTRATIVE EXPERIMENTAL RESULT

The algorithm is implemented using C++. In order to illustrate the result, we considered a simple scenario with 5 robots as shown in Figure 6. The robot computing Voronoi cell is labeled q_0. The expansion phase terminated in 3rd circle and out of four chords (L_1, L_2, L_3, L_4), the chord L_2 is discarded and it can be seen that remaining chords form a closed convex polygon shown with dark lines. In this situation, the polygon obtained after expansion phase itself is the final Voronoi cell. Thus, contraction phase does not modify the polygon created.

V. RELATED WORK

Computation of the complete Voronoi partition is a standard problem addressed in computational geometry [6]. Calculation of the Voronoi partition requires the underlying communication graph of the nodes to be connected. There are only a few existing techniques that employ a distributed computation of the Voronoi cell. In [7], an approximate Voronoi cell is constructed for each node using its four closest nodes, one from each quadrant. A filter-and-refine algorithm is presented in [8], where in the first phase, the sensor node computes an approximate Voronoi cell based on the nodes within its radio range, which is refined by communicating with other nodes within an impact range. In [5], the authors consider a bounded region and an initial node set as a subset of the entire node set that yields a bounded Voronoi cell. Then a geographic routing protocol called GPSR is used to probe for nodes that reduce the initial Voronoi cell and refine it. A similar approach is used in [9], where the sensors cooperate to refine the Voronoi cell and achieve a faster convergence. The first phase of these algorithms constructs approximate candidate Voronoi cells based on a small number of nodes while using a 'brute force' approach [8]. In contrast to these brute force methods, we use a more systematic approach in a structured, expansion phase to construct the initial, approximate Voronoi cells. Also, the existing algorithms rely on communication protocols to exchange positional information on demand. In contrast our work requires robots to exchange positional information with each other only at the beginning of the algorithm. Finally, most of the existing techniques were proposed for sensor networks and rely on specific communication protocols (e.g., GPSR[5]). On the other hand, our work is also suitable for multi-agent/robot settings and does not rely on any specific communication protocol.

VI. CONCLUSIONS AND FUTURE WORK

We presented a distributed algorithm for computation of exact Voronoi cell in the context of multi-robotic systems. Each robot computing the corresponding Voronoi cell computes the configuration of remaining robots in the polar coordinate system using its position as the reference. The Voronoi cell is computed in two distinct phases, namely expansion phase and contraction phase. It was shown that the proposed algorithm successfully computes the exact Voronoi cell. A detailed analysis of the computational complexity of the proposed algorithm, comparison of its performance with existing centralized and distributed algorithms, and optimization of the algorithm to improve its performance, are some of the ongoing works. We are also working on a distributed algorithm for computation of Voronoi cells constrained by the sensor range of robots, as an extension of the proposed algorithm.

REFERENCES

[1] J. Cortes, S. Martinez, T. Karata, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.
[2] A.A. Khalek, L. Al-kanj, Z. Dawy, and G. Turkiyyah, "Site placement and site selection algorithms for umts radio planning with quality constraints," in *Proc of IEEE 17th conf on telecommunication*, 2010, pp. 375–381.
[3] H.-J. Lee, Y.-H. Kim, Y.-H. Han, and C.Y. Park, "Centroid-based movement assisted sensor deployment schemes in wireless sensor networks," in *proc of IEEE Vehicular Technology Conference*, 2009, pp. 1–5.
[4] I. Stanoi, M. Riedwald, D. Agrawal, and A.E. Abbadi, "Discovery of influence sets in frequently updtaed databases," in *Proc of 27th International Conference on Very Large Data Bases*, 2001, pp. 99–108.
[5] B.A. Bash and P.J. Desnoyers, "Exact distributed voronoi cell computation in sensor networks," in *Proc of the Sixth IEEE/ACM Conference On Information Processing in Sensor Networks*, 2007, pp. 236–243.
[6] F. Aurenhammer, "Voronoi diagrams - a survey of a fundamental geometric data structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345–405, 1991.
[7] M. Sharifzadeh and C. Shahabi, "Supporting spatial aggregation in sensor network databases," in *Proc of 12th International Symposium of ACM GIS*, 2004.
[8] B. Harrington and Y. Huang, "In-network surface simplification for sensor fields," in *Proc of the 13th Annual ACM International Workshop on GIS*, 2005.
[9] Y. N. Rodríguez, H. Xiao, K. Islam, and W. Alsailih, "A distributed algorithm for computing voronoi diagram in the unit disk graph model," in *Proc of 20th Canadian Conference in Computational Geometry*, 2008.