Sensor Placement based on Delaunay Triangulation for Complete Confident Information Coverage in An Area with Obstacles

Lu Dai and Bang Wang
School of Electronic, Information and Communications,
Huazhong University of Science and Technology (HUST), Wuhan, China.
email: wangbang@hust.edu.cn

***Abstract*—This paper studies the sensor placement problem for ensuring complete coverage in an area with obstacles. Instead of using the simplistic disk coverage model, we adopt our recently proposed confident information coverage model for field attribute monitoring applications. We propose a node placement algorithm based on iterative Delaunay triangulation, which is to first obtain Delaunay triangles for some initial seed nodes. Among all Delaunay triangles, we propose algorithms to find a valid one yet with the largest coverage hole for placing a new node. The Delaunay triangulation process is then repeated, until all the Delaunday triangles can be completely covered. Simulation results show that our algorithm has comparable performance in terms of the number of placed nodes, compared with a peer algorithm based on a grid approach to discretize the continuous field. However, our algorithm can truly achieve complete coverage yet with significantly smaller computation time.**

I. Introduction

A wireless sensor network (WSN) consisting of a large number of low cost sensor nodes has many practical applications [1]. The basic functionality of a sensor node is to sense the environment by converting physical stimulus into recordable signals. For example, in precision agriculture, a sensor network can be deployed in a farm to monitor and collect its spatial data, like the soil temperature, humidity and fertility [2] [3]. With such spatial data, we can improve the efficiency of soil irrigation and fertilization so as to increase the crop production.

How to efficiently deploy a sensor network is one of the most important design issues in WSNs. Generally, existing deployment strategies can be classified into the random deployment and deterministic placement [4]. In random deployments, sensor nodes are randomly scattered into the field of interests; While in deterministic placements, nodes can be deployed at desired locations. Many node deployment strategies have been proposed in the literature [5] [6]. Although they have different assumptions and constraints, almost all of these deployment techniques have adopted the *network coverage* as a key performance metric for measuring deployment quality.

Network coverage is closely dependent on the adopted coverage model, which is used to characterize the individual sensor's sensing capability and quality. Many coverage models have been proposed for different types of sensors and applications [4]. For example, the widely used disk coverage model assumes that a sensor can cover a disk centered at itself with radius of its sensing range. However, the disk model is too simplistic, which does not consider the spatial correlation of physical phenomenon and the information processing via sensor collaboration. Motivated from the applications of precision agriculture, we have proposed a new sensor coverage model, called *confident information coverage* (CIC, or Φ-coverage), based on the theory of field reconstruction [7].

In this paper, we study the problem of sensor placement for complete Φ-coverage in an area with obstacles. We note that a sensor field containing obstacles is not uncommon. For example, consider a large farm containing an irregular pond. In such scenarios, it is not possible to place nodes within an obstacle, and it is also not necessary to provide coverage for an obstacle. We propose a node placement algorithm to ensure complete Φ-coverage for the valid region of a sensor field with obstacles. The algorithm is called *greedy placement based on iterative Delaunay triangulation* (GPIDT), which first places some initial nodes on the boundary of the sensor field and obstacle vertices. It then performs a Delaunay triangulation for the initial placement. Among all Delaunday triangles, we propose algorithms to find a valid Delaunay triangle yet with the largest coverage hole. A new sensor node is then placed on this triangle, and we redo the Delaunay triangulation again. The iterative placement is terminated if all the Delaunay triangles can be completely Φ-covered. We compare our algorithm with a peer algorithm [8], which applies a grid approach to discretize the continuous field. Simulation results show that our algorithm is comparable to the grid-based placement [8] in terms of the number of placed nodes. Furthermore, it can truly achieve 100% complete coverage and its computation time is much smaller than this peer algorithm.

The paper is organized as follows. Section II briefly reviews the related work, and Section III introduces some preliminaries on CIC model and Delaunay triangulation. Our GPIDT algorithm is presented in Section IV and examined via simulations in Section V. Finally, Section VI concludes the paper.

II. Related Work

In this section, we briefly review the most related work on sensor deployment. We refer the reader to the comprehensive survey papers for more detailed discussions [4]–[6].

Grid-based placement strategies: In a continuous field, the candidate locations for placing sensors actually are infinite. The grid-based placements first divide the continuous field into consecutive yet countable grid cells, such that a sensor can only be placed at a cell center and complete coverage is achieved for all grid vertices being covered [8]–[13]. For example, Ke et al. prove that the problem of deploying sensors on grid vertices to fully covers critical grid centers using minimum sensors is NP-Complete in [10] and propose an approximation algorithm to solve this problem in [11]. In [8], [13], greedy heuristic algorithms are proposed for achieving confident information coverage based on the grid division. Although the grid-based approach can tackle different areas with or without obstacles, it introduces very high computation complexity, since the number of grid vertices has to be set large enough so as to represent the continuous field.

Pattern-based placement strategies: Based on the disk coverage model, Kershner [14] has proven the optimal placement pattern in an infinite plane; While recently Wang et al. [15] prove the optimal placement pattern in long belt scenarios. Sensors can be placed on the vertices of an optimal pattern so as to minimize the number of sensors to be deployed. For constrained areas with obstacles, a common approach is to divide the areas into different parts. For some large parts, sensors are placed according to the placement pattern; While for some small parts that contain the boundaries of obstacles, sensors can be placed one-by-one to eliminate coverage holes [16]–[18]. The pattern-based placements greatly reduce the computation complexity, since the pattern vertices are much fewer than the grid vertices. However, these strategies are all based on the properties of the disk coverage model, which may not be directly applicable to the confident information coverage.

Computational geometry-based strategies: Computational geometry as a mathematical tool mainly deals with various computational problems of geometric nature, such as the well-known Art Gallery Problem. For sensor placement problems, the most related technique is the Delaunay triangulation [19]–[23]. For example, For the disk coverage model, Qiu and Shen [19] provide a method to determine the complete coverage of a triangle. Wu et al. [20] propose a two-phase sensor deployment via iterative Delaunay Triangulation. It first evenly places sensors along the contour lines of the boundaries and obstacles. Based on the deployed sensors, it then applies the Delaunay triangulation to identify the largest coverage hole to place a new sensor. Derr and Manic [22], [23] apply a similar procedure, yet including a more sophisticated node removal and triangulation smoothing process. Compared with the grid-based approaches, its computation complexity can be much reduced, and compared with the pattern-based approaches, it can well handle various area and obstacle shapes.

In this paper, we also adopt the computational geometry strategy and the Delaunay Triangulation technique, but based on the new confident information coverage model. To the best of our knowledge, we are the first one studying this topic.

III. Preliminaries

A. Confident Information Coverage

The CIC model is based on the theory of field reconstruction. Sensors are deployed within the sensor field to sample the attribute of some physical phenomena, and field reconstruction is to use their sampling values to interpolate or estimate the physical attribute for those unsampled locations.

For each space point x, we only use the samplings of those sensors located within its *correlation range* denoted by D for its reconstruction. Furthermore, we use its time-average *root mean square error* (RMSE), denoted by $\Phi(x)$, to evaluate its reconstruction quality. The space point x is Φ-covered, if $\Phi(x)$ is not larger than the application requirement ϵ, i.e., $\Phi(x) \leq \epsilon$. A sensor field is said being completely Φ-covered, if all the space points within the field are Φ-covered.

In most applications, the physical phenomenon can be assumed as a second-order stationary Gaussian process [24], and its spatial statistics of the degree of spatial dependence can be described by a *variogram* function $\gamma(h)$:

$$\gamma(h) = 1 - e^{-\frac{h^2}{\alpha^2}}, \tag{1}$$

where h is the Euclidean distance between two points, and the constant α is related to the correlation range $D = \sqrt{3}\alpha$.

One of the widely used spatial reconstruction technique is the ordinary Kriging technique [25], [26]. Let $S(x) \equiv \{s_1, ..., s_n\}$ denote the set of sensors located within the correlation range of the point x. Then based on the ordinary Kridging and after some algebra, we can compute $\Phi(x)$ by

$$\Phi(x) = \sqrt{\Upsilon^T \mathbf{K}^{-1} \Upsilon}, \tag{2}$$

where

$$\mathbf{K} = \begin{pmatrix} \gamma(s_1, s_1) & \gamma(s_1, s_2) & \dots & \gamma(s_1, s_n) & 1 \\ \gamma(s_2, s_1) & \gamma(s_2, s_2) & \dots & \gamma(s_2, s_n) & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \gamma(s_n, s_1) & \gamma(s_n, s_2) & \dots & \gamma(s_n, s_n) & 1 \\ 1 & 1 & \dots & 1 & 0 \end{pmatrix}, \tag{3}$$

and

$$\Upsilon = (\gamma(s_1, x), \gamma(s_2, x), ..., \gamma(s_n, x), 1)^T . \tag{4}$$

Eq. (2) indicates that $\Phi(x)$ depends on not only the spatial correlations between x and the sensors within $S(x)$, but also the spatial correlations in between the sensors within $S(x)$.

B. Delaunay Triangulation

Given a point set $\mathcal{S}$ on a plane, the triangulation of $\mathcal{S}$ is defined as the planar subdivision whose bounded faces are triangles and whose vertices are the points in $\mathcal{S}$. A triangulation for $\mathcal{S}$ in a plane is a Delaunay triangulation, if no point in $\mathcal{S}$ is inside the circumcircle of any triangle [27]. This is also called the empty circle property. Note that all these Delaunay triangles formed by three points in $\mathcal{S}$ do not overlap with each other. If each Delaunay triangle can be completely covered by three sensors at the triangle vertices, the polygon consisting of these Delaunay triangles can be completely covered. If the field of interests is a bounded polygon, a simple approach for its complete coverage is to place some sensors on the polygon vertices and edges. In this paper, we provide the computation method for the complete Φ-coverage of a triangle.

C. Problem Description

We study the sensor placement problem in a sensor field with obstacles. Within the sensor field, there may exist one or more obstacles, each modeled by either a convex polygon or a concave polygon. For each obstacle polygon, it is not possible to place a sensor within the polygon, but we can place sensors on its boundary (i.e., the polygon sides and vertices). Furthermore, for each polygon, it is not necessary to cover its inner area and its boundary. For ease of presentation, we assume that the boundary of an obstacle polygon belongs to its inner area too. Therefore, we call the *valid region* of the sensor field as the region within the sensor field containing no inner area and no boundary of any obstacle polygon.

Our sensor placement problem is to place the least number of sensors to completely Φ-cover the valid region of such a sensor field. Due to the existence of irregular obstacles, the pattern-based placement strategies are not directly applicable. Although the grid-based placement strategies, such as the algorithms proposed in our previous work [8], can be used, they are still a kind of approximation approaches. As to be shown in our simulation in Section V, they cannot provide the truly 100% complete coverage, and their computation complexity is very high. In the rest of this paper, we propose a computational geometry-based placement algorithm to provide truly 100% complete coverage, yet with significantly small computation time.

IV. Greedy Placement based on Iterative Delaunay Triangulation

In this section, we propose a *greedy placement based on iterative Delaunay triangulation* (GPIDT) to iteratively place one sensor at each step, until the complete Φ-coverage of the sensor field can be satisfied. Algorithm 1 presents the psudo-codes of the proposed GPIDT algorithm. The input of the algorithm includes the vertex set of the field boundary polygon V_f and the vertex set of the obstacle boundary polygon V_o. The output of the algorithm is the coordinate set of the placed sensors S.

The GPIDT algorithm initially places some seed nodes (line 1), and performs a Delaunay triangulation to obtain the set of Delaunay triangles (line 2). Note that each Delaunay triangle (DT) is formed by three deployed nodes. The function placeSeedNode ensures that the boundary part of the field polygon can be completely Φ-covered. The **while** loop is to ensure that all the *valid* DTs by the deployed nodes S are completely Φ-covered, by which all the valid area within the polygon formed by the boundary seed nodes can be completely Φ-covered. A *valid* DT is defined as such a DT that is not completely located within an obstacle.

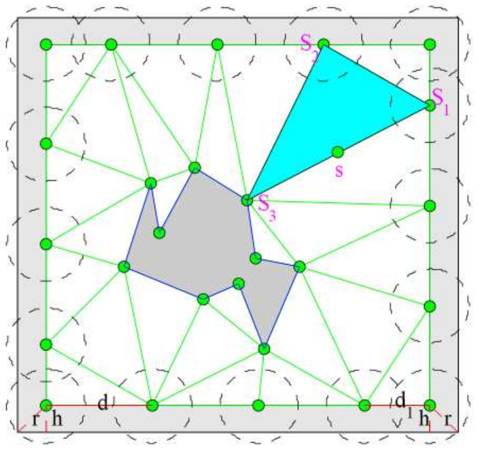
Algorithm 1 The GPIDT Algorithm

```
Input: $V_f$-the vertex set of the field boundary polygon;
       $V_o$-the vertex set of the obstacle boundary polygon;
Output: $S$ - the coordinate set of placed sensor nodes;
1: $S$ = placeSeedNode($V_f, V_o$);
2: $\Delta$ = delaunayTriangulation($S$);
3: while $\Delta \neq \emptyset$ do
4:   $R^* = 0$; $t^*$ = NULL;
5:   for each $t \in \Delta$ do
6:     if isValidTriangle(t)=TRUE then
7:       if isCICCovered(t)=FALSE then
8:         $R$ = computeCircumcircleRadius($t$);
9:         if $R > R^*$ then
10:          $R^* = R$; $t^* = t$;
11:        end if
12:      end if
13:    end if
14:  end for
15:  if $t^* \neq$ NULL then
16:    $s$ = placeOneNode($t^*$);
17:    $S = S \cup \{s\}$;
18:    $\Delta$ = delaunayTriangulation($S$);
19:  else
20:    $\Delta = \emptyset$; break;
21:  end if
22: end while
23: return $S$;
```

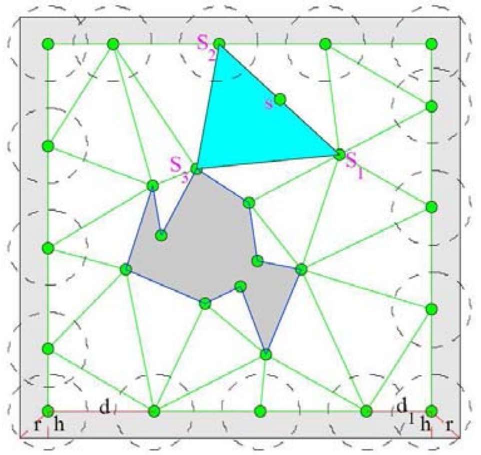
In each iteration of the **while** loop, we decide, if necessary, where to place one new node as follows: For each DT, we first examine whether it is a valid DT. Then for each valid DT, we next examine whether it can be completely Φ-covered. If a valid DT cannot be completely Φ-covered, we compute its circumcircle radius (line 8), which is used to approximately measure the size of its coverage hole. After the **for** loop, an uncovered DT with the largest circumcircle radius is found, recorded by t^*. If there exists an t^*, we then place a new node on t^* (line 16), redo the Dalaunay triangulation (line 18), and move to the next iteration of the **while** loop. If t^* does not exist, which means all the valid DTs can be completely Φ-covered, then the **while** loop is terminated (line 20).

Fig. 1 illustrates the initial seed node placement and the first and second Delaunay triangulation. Note that the Φ-coverage allows one or more sensors to cover a space point. In Fig. 1, a dashed circle represents the sensing disk coverage only by using one sensor. We provide complete coverage for the boundary area (colored by the light grey) by using one-sensor or two-sensor Φ-coverage. And we use the three-sensor Φ-coverage to completely cover a valid Delaunay triangle. More details will be explained in the following subsections.

In the proposed GPIDT algorithm, we use Matlab function delaunary to implement the delaundayTriangulation function. Let a, b, c denote the side length of a triangle. The



(a) Initial seed node placement and the first Delaunday triangulation. The valid DT with the largest coverage hole is $\triangle \mathrm{S_1S_2S_3}$, and a new node s is placed on the midpoint of its longest side $\overline{\mathrm{S_1S_3}}$.



(b) The second Delaunday triangulation. The valid DT with the largest coverage hole is $\triangle \mathrm{S_1S_2S_3}$, and a new node s is placed on the midpoint of its longest side $\overline{\mathrm{S_1S_2}}$.

Fig. 1. Illustration of the initial seed node placement and the first and second iteration of Delaunday triangulation.

circumcircle radius R of this triangle is computed by

$$R = \frac{abc}{\sqrt{(a+b+c)(a+b-c)(a-b+c)(-a+b+c)}}. \quad (5)$$

We next introduce our implementation of the other four customized functions.

A. The *isCICCovered* function

The function isCICCovered examines whether a triangle can be completely Φ-covered by the three sensors located at its vertices. At first, we examine whether a triangle is illegal for such examination. Recall that only those sensors within the correlation range of a space point are considered for its Φ-coverage computation. Therefore, the length of an arbitrary line segment in a triangle should be no larger than the correlation range D. The longest line segment in a triangle is its longest side. Furthermore, all the triangle sides are chords of its circumcircle, and the longest chord of a circle is the circle diameter. To ensure Φ-coverage computation for a triangle, the circumcircle diameter should be no larger than the correlation range. That is, $R \le \frac{D}{2}$, and we call such a DT a *legal* triangle in this paper. Note that if a DT is an illegal triangle, it will not pass the complete Φ-coverage test.

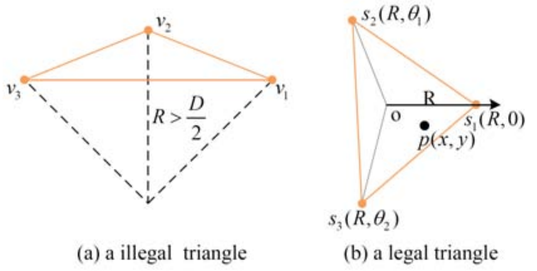


Fig. 2. (a) An illegal triangle with circumcircle radius $R > \frac{D}{2}$. (b) A legal triangle with circumcircle radius $R \le \frac{D}{2}$, and its polar coordinate system.

For a legal triangle $\triangle s_1 s_2 s_3$, we use its circumcircle center to build a polar coordinate system, and set the polar coordinate for the three vertices as: $s_1(R,0)$, $s_2(R,\theta_1)$ and $s_3(R,\theta_2)$. Note that in the corresponding cartesian coordinate system, we have $s_1(R,0)$, $s_2(R\cos\theta_1, R\sin\theta_1)$ and $s_3(R\cos\theta_2, R\sin\theta_2)$. Let $p(x,y)$ denote an arbitrary point within the triangle or on the sides of the triangle. Then it should meet the following constraint:

$$\begin{cases} \overrightarrow{s_1s_2} \times \overrightarrow{s_1p} \ge 0 \\ \overrightarrow{s_2s_3} \times \overrightarrow{s_2p} \ge 0 \\ \overrightarrow{s_3s_1} \times \overrightarrow{s_3p} \ge 0 \end{cases} \quad (6)$$

Then the constraints about (x,y) can be calculated by

$$\begin{pmatrix} \cos\frac{\theta_1}{2} & \sin\frac{\theta_1}{2} \\ -\cos\frac{\theta_2}{2} & -\sin\frac{\theta_2}{2} \\ \sin\theta_2 - \sin\theta_1 & \cos\theta_1 - \cos\theta_2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \le \begin{pmatrix} R\cos\frac{\theta_1}{2} \\ -R\cos\frac{\theta_2}{2} \\ R\sin(\theta_2-\theta_1) \end{pmatrix} \quad (7)$$

The distances in between sensors can be computed by:

$$d(s_1,s_2) = 2R\left|\sin\frac{\theta_1}{2}\right|, \quad d(s_1,s_3) = 2R\left|\sin\frac{\theta_2}{2}\right|,$$

$$d(s_2,s_3) = 2R\left|\sin\frac{\theta_2-\theta_1}{2}\right|,$$

and the distances between the point p and the sensors can be computed by

$$d(p,s_1) = \sqrt{x^2+y^2-2Rx+R^2},$$
$$d(p,s_2) = \sqrt{x^2+y^2-2R(x\cos\theta_1+y\sin\theta_1)+R^2},$$
$$d(p,s_3) = \sqrt{x^2+y^2-2R(x\cos\theta_2+y\sin\theta_2)+R^2}.$$

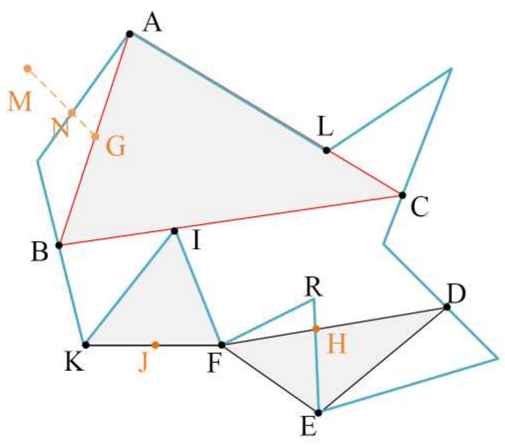


Fig. 3. Illustration of determine a valid DT, where △KIF and △DEF are valid DTs and △ABC is not a valid DT.

Let A,B,C,E,F,G denote the variogram functions of the above distances, we have

$$
\begin{aligned}
A &= \gamma(s_1, s_2) = 1 - e^{\frac{-4R^2 \sin^2 \frac{\theta_1}{2}}{\alpha^2}}, \\
B &= \gamma(s_1, s_3) = 1 - e^{\frac{-4R^2 \sin^2 \frac{\theta_2}{2}}{\alpha^2}}, \\
C &= \gamma(s_2, s_3) = 1 - e^{\frac{-4R^2 \sin^2 \frac{\theta_2-\theta_1}{2}}{\alpha^2}}, \\
E &= \gamma(p, s_1) = 1 - e^{-\frac{x^2+y^2-2Rx+R^2}{\alpha^2}}, \\
F &= \gamma(p, s_2) = 1 - e^{-\frac{x^2+y^2-2R(x\cos\theta_1+y\sin\theta_1)+R^2}{\alpha^2}}, \\
G &= \gamma(p, s_3) = 1 - e^{-\frac{x^2+y^2-2R(x\cos\theta_2+y\sin\theta_2)+R^2}{\alpha^2}}.
\end{aligned}
$$

So the RMSE of the point $p(x, y)$ can be computed by

$$
\Phi(p) = \sqrt{\frac{2 \times (-ABC + L(p))}{(A+B+C)^2 - 2(A^2+B^2+C^2)}}, \qquad (8)
$$

where $L(p) = -AG^2 - BF^2 - CE^2 + (-A+B+C)(AG+EF) + (A-B+C)(BF+EG) + (A+B-C)(CE+FG)$.

With the help of the Matlab fmincon function, the maximum of $L(p)$, as well as that of $\Phi(p)$, can be computed. The triangle is said being completely confident information covered, if the maximum of $\Phi(p)$ is not larger than the application requirement ϵ, i.e., $\Phi(p) \le \epsilon$.

B. The *isValidTriangle* function

The isValidTriangle function determines whether a DT is a valid one. The *valid region* of the sensor field is defined as a part of the sensor field not within any obstacle. A valid DT contains at least some valid region. Therefore, if any of the vertex of a triangle is within the valid region, then this triangle is a valid one. Furthermore, as we do not place any sensor within an obstacle, we only need to consider those triangles with all vertices on the sides or vertices of an obstacle. A triangle with all vertices on the obstacle boundary is called a *testing triangle*.

The basic idea of the isValidTriangle function is to examine whether each side of a testing triangle is completely located within an obstacle. If any side of a testing triangle is not completely within an obstacle, then it is a valid triangle. Otherwise, it is not a valid one [27]. As shown in Fig. 3, the testing triangle △FIK and △DEF are valid DTs; While △ABC is not a valid DT, and all of its sides are within the obstacle.

We next present how to determine whether a line segment $\overline{\mathrm{uv}}$ is completely within an obstacle polygon. Note that $\overline{\mathrm{uv}}$ is also the side of a Delaunay triangle and its endpoints u and v are each placed with one sensor. So u and v can only be either the polygon vertices or located on some side of the polygon. If the line segment $\overline{\mathrm{uv}}$ coincides with one polygon side, we simply determine that it is within the obstacle. For example, in Fig. 3 the DT side $\overline{\mathrm{IK}}$ is considered as within the polygon.

At first, for each polygon side $\overline{\mathrm{pq}}$, we compute the intersection point between $\overline{\mathrm{uv}}$ and $\overline{\mathrm{pq}}$. Note that due to its endpoint property, $\overline{\mathrm{uv}}$ must intersect with at least one polygon side, or a part of $\overline{\mathrm{uv}}$ coincides with one polygon side. In the latter case, we only include the polygon vertices on the line segment as its intersection points. For example, the line segment $\overline{\mathrm{AC}}$ has a part of $\overline{\mathrm{AL}}$ coincides with the polygon side, and we only include the polygon vertex A and L as its intersection point.

Let z denote one of such intersection points between a line segment $\overline{\mathrm{uv}}$ and a polygon side $\overline{\mathrm{pq}}$. If z does not superpose with any of the endpoint of $\mathrm{u, v}$ and z is not any other vertex of the polygon, then there must exist some part of the line segment $\overline{\mathrm{uv}}$ is within the valid region. This is because that a polygon side is a division of the valid region and invalid region. For example, in Fig. 3 the intersection point H between the triangle side $\overline{\mathrm{DF}}$ and the polygon side $\overline{\mathrm{ER}}$ is not any endpoint nor polygon vertex. The line segment $\overline{\mathrm{FH}}$ is within the valid region, so $\overline{\mathrm{FD}}$ is not completely within the obstacle.

Now we consider the case that an intersection point z either is one of the endpoint u and v, or is one of the polygon vertex. For example, in Fig. 3 the intersection point L and I such intersection points as they are also the polygon vertices. We then form a point set $\mathcal{P}$ containing the two endpoints of the line segment as well as the such intersection points. And the points in $\mathcal{P}$ are ordered according to the x-coordinate in an increasing order. Let $\mathcal{P} = \{z_1, ..., z_n\}$ denote such a point set. For example, in Fig. 3, $\mathcal{P} = \{I, F\}$ for $\overline{\mathrm{FI}}$, $\mathcal{P} = \{K, F\}$ for $\overline{\mathrm{FK}}$, $\mathcal{P} = \{B, A\}$ for $\overline{\mathrm{AB}}$, $\mathcal{P} = \{A, L, C\}$ for $\overline{\mathrm{AC}}$, and $\mathcal{P} = \{B, I, C\}$ for $\overline{\mathrm{BC}}$.

For each line segment $\overline{\mathrm{z_i z_{i+1}}}$ $(i = 1, ..., n-1)$ in the point set $\mathcal{P}$, we check whether or not its midpoint $\mathrm{z_i^m}$ is within the polygon. If all such $n-1$ midpoints are within the polygon, then we conclude that the line segment $\overline{\mathrm{uv}}$ is within the polygon; On the contrary, if any of such midpoint is not within the polygon, then we conclude that the line segment $\overline{\mathrm{uv}}$ is not within the polygon.

For example, the only one midpoint of $\mathcal{P} = \{I, F\}$ for $\overline{\mathrm{IF}}$ is within the boundary of the polygon (which we also consider to be within the polygon), then $\overline{\mathrm{IF}}$ is within the polygon. And so is for $\overline{\mathrm{AB}}$. The only one midpoint of $\mathcal{P} = \{K, F\}$ is within

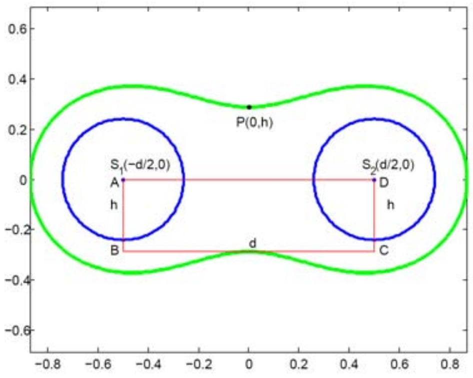


Fig. 4. Illustration of one-sensor Φ-coverage by the blue circled disk area and two-sensor Φ-coverage by the green dumbbell area.

the valid area, so the $\overline{\mathrm{KF}}$ is not completely within the polygon. Since the midpoint of $\overline{\mathrm{BI}}$ and the midpoint of $\overline{\mathrm{IC}}$ are within the polygon, the line segment $\overline{\mathrm{BC}}$ is completely within the polygon. For the line segment $\overline{\mathrm{AC}}$, since $\overline{\mathrm{AL}}$ coincides with the polygon side, it is considered to be within the polygon. And the midpoint of $\overline{\mathrm{LC}}$ is within the polygon, so the line segment $\overline{\mathrm{AC}}$ is within the polygon.

If $\overline{\mathrm{z_i z_{i+1}}}$ coincides with the a polygon side, such as the line segment $\overline{\mathrm{AL}}$, we simply regard that it is within the polygon. In what follows, we do not consider such a superposition case. We use the *contradiction method* to prove that if the midpoint of $\overline{\mathrm{z_i z_{i+1}}}$ is within the polygon, then $\overline{\mathrm{z_i z_{i+1}}}$ is also within the polygon. Recall that by the definition of $\mathcal{P}$, there does not exist any intersection point between the $\overline{\mathrm{z_i z_{i+1}}}$ and any of polygon side. We take the line segment $\overline{\mathrm{AB}}$ as an example of $\overline{\mathrm{z_i z_{i+1}}}$, which should not contain any other intersection point between the line segment $\overline{\mathrm{AB}}$ and the polygon. If the midpoint of G is within the polygon and $\overline{\mathrm{AB}}$ is not completely within the polygon, then there must exist a point M on $\overline{\mathrm{AB}}$ outside the polygon. Let us connect the two points G and M. Since a part of the line segment $\overline{\mathrm{MG}}$ is not within the polygon, this segment must have an intersection with one of the polygon side. Let N denote such an intersection point. However, the existence of N is contradict with our definition that the point set $\mathcal{P}$ is a complete set containing all intersection points of a line segment $\overline{\mathrm{uv}}$ with the polygon. That is, N should be included into the point set $\mathcal{P}$ of $\overline{\mathrm{AB}}$, and divide $\overline{\mathrm{AB}}$ into two segments. But this is not true, so $\overline{\mathrm{AB}}$ is within the the polygon.

C. The placeSeedNode function

The placeSeedNode function is initially place two types of seed nodes. One type is to provide complete Φ-coverage for the sensor field boundary region; The other type is to put some seed nodes at each vertex of the inner obstacle polygon, so as to help generating an initial Delaunay triangulation for the whole sensor field.

We next present our implementation of placing the first type of seed nodes. Notice that the Φ-coverage of a single sensor is actually the same as the disk model [7]. Furthermore, the radius of the disk can be computed by

$$r = \alpha\sqrt{-\ln(1-\frac{\epsilon^2}{2})}. \quad (9)$$

The Φ-coverage of two sensors is also dependent on their distance. In [28], it has been proven that the Φ-coverage of two sensors is a dumbbell shape, if the distance between them is less than a some threshold. Fig. 4 illustrates the Φ-coverage of one sensor by the two blue disks and the Φ-coverage of two sensors by the green dumbbell.

Instead of placing seed nodes on the field boundary, we propose to place seed nodes on its *contour* to completely Φ-cover some *offset area* along with the boundary. For a boundary vertex, we use the Φ-coverage of one sensor. That is, a sensor is placed on its angle bisector, with the distance to the vertex equal to the disk radius r. Then we can compute the height of this sensor to the boundary side, denoted by h. For a boundary side, we use the Φ-coverage of two sensors. As shown in Fig. 4, to provide complete Φ-coverage, we need to ensure that the saddle-point p is on the boundary, by which we can compute the distance between the two sensors as follows. Let $\Phi(p) = \epsilon$, and for the example coordinate system in Fig. 4, we have

$$h = \alpha\sqrt{\ln(4) - \frac{d^2}{4\alpha^2} - \ln(3 - 2\epsilon^2 + e^{-\frac{d^2}{\alpha^2}})}. \quad (10)$$

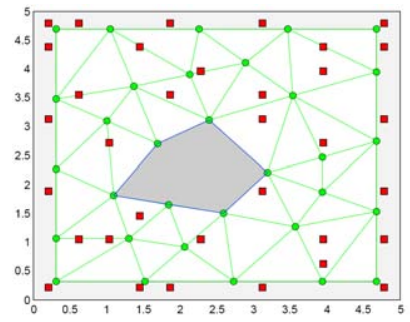
Fig. 1 illustrates the initial seed node placement on a contour line. We cover a boundary vertex by using the one-sensor Φ-coverage. By setting the distance to a boundary vertex equal to r, we can compute the parameter h, and then the inter-node distance d as well.

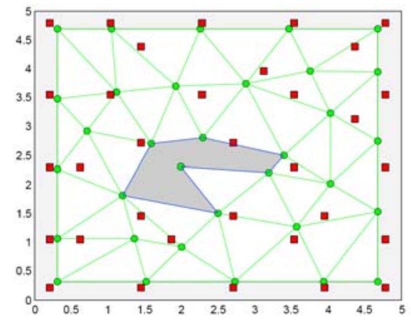
D. The placeOneNode function

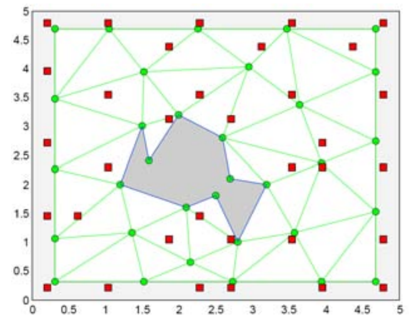
The placeOneNode function is to place one new node on the largest uncovered DT so as to reduce some uncovered area and to prepare the next Delaunay triangulation. The function is implemented as follows. We first consider the midpoint of the largest side. If this midpoint is within the valid region, we then place a new sensor on it. If this midpoint is within the obstacle, we next consider the midpoint of the second-largest side. If all the three sides' midpoints are within the obstacle, we turn to the intersection point between a triangle side and obstacle polygon side. Take the triangle $\triangle\mathrm{DEF}$ in Fig. 3 for example. The midpoint of the largest side $\overline{\mathrm{DF}}$ and the midpoint of second largest side $\overline{\mathrm{DE}}$ are both within the obstacle. We then put the new sensor on the midpoint of $\overline{\mathrm{EF}}$.

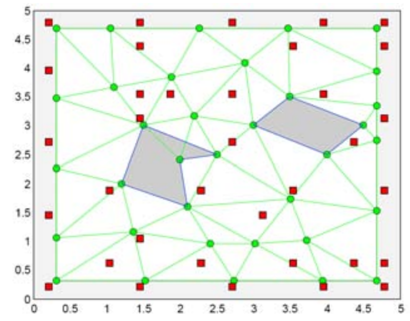
V. Simulation Results

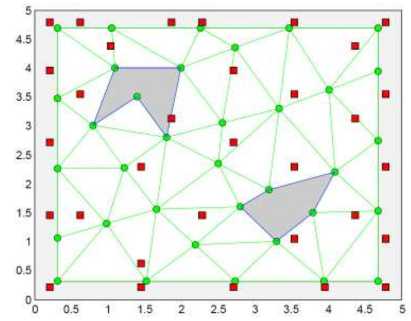
We consider a sensor field of 5×5 and set $\epsilon = 0.6$ and $D = \sqrt{3}$ for Φ-coverage. We compare our algorithm with the another node placement algorithm, the *candidate location based greedy algorithm* (CLBGA), for Φ-coverage [8]. The CLBGA approximates the continuous field coverage with the discrete grid coverage. It first discretizes the continuous sensor field into many equal grid cells. A sensor can only be placed at some grid cell center, which is called a candidate location; and the complete field Φ-coverage is converted into Φ-covering all


(a) GPIDT: $N = 32$ and $CR = 100\%$; CLBGA (12×12): $N = 34$ and $CR = 97.95\%$


(b) GPIDT: $N = 33$ and $CR = 100\%$; CLBGA (12×12): $N = 33$ and $CR = 99.59\%$


(c) GPIDT: $N = 33$ and $CR = 100\%$; CLBGA (12×12): $N = 34$ and $CR = 98.63\%$


(d) GPIDT: $N = 35$ and $CR = 100\%$; CLBGA (12×12): $N = 35$ and $CR = 98.88\%$


(e) GPIDT: $N = 35$ and $CR = 100\%$; CLBGA (12×12): $N = 32$ and $CR = 98.55\%$

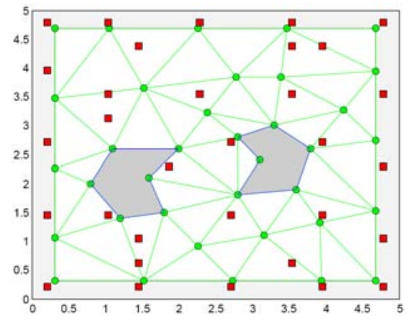

(f) GPIDT: $N = 36$ and $CR = 100\%$; CLBGA (12×12): $N = 32$ and $CR = 98.43\%$

Fig. 5. Realization of the node placement by GPIDT (green dots and corresponding Delaunay triangulation), and by CLBGA algorithm (red squares).

TABLE I
COMPARISON OF THE GPIDT AND CLBGA ALGORITHM.

		Field (a)				Field (b)				Field (c)			
		N	σ_N	CR (%)	T(s)	N	σ_N	CR (%)	T(s)	N	σ_N	CR (%)	T
GPIDT		32	0	100%	3.5	33	0	100%	3.4	33	0	100%	2.6
CLBGA	9×9	29.6	1.34	98.97%	443.6	31.0	1.41	99.17%	541.4	29.4	1.14	97.46%	485.0
	10×10	37.2	1.64	99.86%	1002.0	38.0	1.22	99.50%	1115.8	37.2	1.64	98.63%	973.0
	11×11	29.6	1.14	97.63%	1899.4	31.4	1.14	97.56%	1960.8	32.4	1.52	97.77%	2261.7
	12×12	33.6	0.55	98.54%	2090.6	34.2	1.10	99.19%	2598.0	34.2	0.45	98.78%	3055.0
	13×13	32.2	1.30	98.15%	3822.4	32.4	0.89	97.66%	5334.5	31.6	2.41	98.73%	4090.2
		Field (d)				Field (e)				Field (f)			
		N	σ_N	CR (%)	T(s)	N	σ_N	CR (%)	T(s)	N	σ_N	CR (%)	T
GPIDT		35	0	100%	4.0	35	0	100%	3.7	36	0	100%	4.5
CLBGA	9×9	31.4	1.52	99.25%	717.4	28.6	0.89	95.49%	636.2	28.0	1.22	96.27%	708.2
	10×10	37.0	1.41	98.76%	1587.0	36.4	0.55	99.59%	1534.5	37.4	1.52	99.39%	1654.3
	11×11	31.0	1.41	96.74%	3044.8	31.4	0.89	96.65%	3337.3	31.4	2.41	97.09%	3021.5
	12×12	34.2	1.30	99.06%	4770.2	34.0	1.22	99.00%	4714.5	33.4	3.36	98.59%	3943.5
	13×13	32.4	2.19	98.09%	6730.8	31.4	2.61	99.10%	9811.0	33.4	0.89	97.98%	5951.2

the grid vertices. The basic idea of CLBGA is to greedily place one sensor each time to minimize the number of uncovered grid vertices. In our CLBGA implementation, for grid vertices and grid centers located within an obstacle, we do not consider the Φ-coverage of such grid vertices and do not place nodes on such grid centers.

Fig. 5 plots the realization of the sensor placement by the GPIDT and CLBGA algorithm for six areas with different types and shapes of obstacles. TABLE I compares the GPIDT with the CLBGA algorithm on the number of deployed nodes N, the standard deviation of N, σ_N, the coverage ratio CR and the computation time T. Note that the execution of GPIDT leads to only one sensor placement. Yet the execution of CLBGA may generate different placements if we set different initial random seed. This is because in CLBGA several candidate locations may have the same weight to place a new node and ties are broken randomly. So in the table, the average value of CLBGA is obtained over five different initial seeds. Furthermore, for the CLBGA algorithm implementation, we can use different grid granularity, like 9×9, 10×10,..., and 13×13, for a same field. However, we use a much finer grid granularity, 0.05×0.05, to obtain testing points and examine the field coverage ratio for an obtained placement. That is, after obtaining a node placement, we examine the percentage of such testing grid vertices being Φ-covered.

From the table, we first observe that in terms of the number of deployed nodes, our GPIDT is comparable to the CLBGA algorithm. On the other hand, we can observe that our GPIDT always achieves 100% Φ-coverage; While the CLBGA cannot. This is because that our GPIDT algorithm is designed for the continuous sensor field, where each Delaunay triangle should be completely Φ-covered. Due to the grid approximation, the CLBGA places nodes only at a few candidate locations to cover selected grid vertices. Notice that although the CLBGA can achieve very high coverage ratio, it still can only be regarded as an approximation solution, as it cannot achieve 100% Φ-coverage. In contrast, our GPIDT algorithm is more theoretically sound for achieving 100% Φ-coverage.

When comparing the computation complexity, our GPIDT is greatly superior than the CLBGA in terms of much smaller computation time. For the CLBGA algorithm, the finer the grid granularity, the higher of its computation complexity. We also note that for the given field, the computation complexity becomes prohibitive high for a higher grid size. When applied in large sensor fields, the high computation time would become a great hurdle for using the CLBGA, since many more grid points should be created. So in large sensor fields, it is expected that our GPIDT algorithm can find more applications.

VI. Conclusions

In this paper, we have studied the sensor placement problem for achieving 100% confident information coverage and have proposed a new algorithm based on the iterative Delaunay triangulation. In the proposed algorithm, we have solved several problems, including how to determine a Delaunay triangle is a valid one and how to determine its complete Φ-coverage. Simulation results have shown that our algorithm achieves comparable performance with a grid-based placement algorithm, in terms of the number of placed nodes. But our algorithm can ensure 100% coverage and its computation time is much smaller. In our future work, we will take into consideration of the network connectivity when designing new sensor placement algorithms.

Acknowledgement: This paper is supported in part by the National Natural Science Foundation of China (No. 61371141). and the Fundamental Research Funds for the Central Universities (No. HUST2015QN081).

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: a survey,” *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] N. Wang, N. Zhang, and M. Wang, “Wireless sensor in agriculture and food industry-recent development and future perspective,” *Computers and Electronics in Agriculture*, vol. 50, no. 1, pp. 1–14, 2006.

[3] A. Camilli, C. E. Cugnasca, A. M. Saraiva, A. R. Hirakawa, and P. L. Corra, “From wireless sensors to field mapping: Anatomy of an application for precision agriculture,” *Computers and Electronics in Agriculture*, vol. 58, no. 1, pp. 25–36, 2007.

[4] B. Wang, “Coverage problems in sensor networks: A survey,” *Computing Surveys*, vol. 43, no. 4, pp. 1–56, 2011.

[5] D. S. Deif and Y. Gadallah, “Classification of wireless sensor networks deployment techniques,” *IEEE Communications Surveys and Tutorials*, vol. 16, no. 2, pp. 834–855, 2014.

[6] M. R. Senouci, A. Mellouk, K. Asnoune, and F. Bouhidel, “Movement-assisted sensor deployment algorithms: a survey and taxonomy,” *IEEE Communications Surveys and Tutorials (to appear)*, pp. 1–19, 2015.

[7] B. Wang, X. Deng, W. Liu, L. T. Yang, and H.-C. Chao, “Confident information coverage in sensor networks for field reconstruction,” *IEEE Wireless Communications*, vol. 20, no. 6, pp. 74–81, 2013.

[8] J. Zhu and B. Wang, “Sensor placement algorithms for con?dent information coverage in wireless sensor networks,” in *IEEE The 23rd International Conference on Computer Communication and Networks (ICCCN)*, 2014, pp. 1–7.

[9] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho, “Grid coverage for surveillance and target location in distributed sensor networks,” *IEEE Transactions on Computes*, vol. 51, no. 12, pp. 1448–1453, 2002.

[10] W.-C. Ke, B.-H. Liu, and M.-J. Tsai, “Constructing a wireless sensor network to fully cover critical grids by deploying minimum sensors on grid points is np-complete,” *IEEE Transactions on Computes*, vol. 56, no. 5, pp. 710–715, 2007.

[11] ——, “Efficient algorithm for constructing minimum size wireless sensor networks to fully cover critical square grids,” *IEEE Transactions on Wireless Communications*, vol. 10, no. 4, pp. 1154–1164, 2011.

[12] B. Wang, “Sensor placement for complete information coverage in distributed sensor networks,” *Journal of Circuits, Systems, and Computers*, vol. 17, no. 4, pp. 627–636, 2008.

[13] H. Xu, J. Zhu, and B. Wang, “On the deployment of a connected sensor network for confident information coverage,” *MDPI Sensors*, vol. 2015, no. 15, pp. 11 277–11 294, 2015.

[14] R. Kershner, “The number of circles covering a set,” *Americal Journal of Mathematics*, vol. 61, no. 3, pp. 665–671, 1939.

[15] B. Wang, H. Xu, W. Liu, and L. T. Yang, “The optimal node placement for long belt coverage in wireless networks,” *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 587–592, 2015.

[16] Y.-C. Wang, C.-C. Hu, and Y.-C. Tseng, “Efficient placement and dispatch of sensors in a wireless sensor network,” *IEEE Transactions on Mobile Computing*, vol. 7, no. 2, pp. 262–274, 2008.

[17] C.-Y. Chang, J.-P. Sheu, Y.-C. Chen, and S.-W. Chang, “An obstacle-free and power-efficient deployment algorithm for wireless sensor networks,” *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 39, no. 4, pp. 795–806, 2009.

[18] X. Li, G. Fletcher, A. Nayak, and I. Stojmenovic, “Placing sensors for area coverage in a complex environment by a team of robots,” *ACM Transactions on Sensor Networks*, vol. 11, no. 1, pp. 1–22, 2014.

[19] C. Qiu and H. Shen, “A delaunay-based coordinate-free mechanism for full coverage in wireless sensor networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 4, pp. 828–839, 2014.

[20] C.-H. Wu, K.-C. lee, and Y.-C. Chung, “A delaunay triangulation based method for wireless sensor network deployment,” *Computer Communications (Elsevier)*, vol. 30, no. 14-15, pp. 2744–2752, 2007.

[21] H. Tan, Y. Wang, X. Hao, Q.-S. Hua, and F. C. M. Lau, “Arbitrary obstatcles constrained full coverage in wireless sensor networks,” in *The 5th International Conference on Wireless Algorithms, Systems and Applications (WASA)*, 2010, pp. 1–10.

[22] K. Derr and M. Manic, “Wireless sensor network configuration-part i: Mesh simplification for centralized algorithms,” *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3, pp. 1717–1727, 2013.

[23] ——, “Wireless sensor network configuration-part ii: Adaptive coverage for decentralized algorithms,” *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3, pp. 1728–1738, 2013.

[24] R. Webster and M. A.Oliver, *Geostatistics for Environmental Scientists (Second Edition)*. Wiley, 2007.

[25] B. Harrington, Y. Huang, J. Yang, and X. Li, “Energy-efficient map interpolation for sensor fields using kriging,” *IEEE Transactions on Mobile Computing*, vol. 8, no. 5, pp. 622–635, 2009.

[26] G. Hernandez-Penaloza and B. Beferull-Lozano, “Field estimation in wireless sensor networks using distributed kriging,” in *IEEE International Conference on Communications (ICC)*, 2012, pp. 724–729.

[27] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications (Third Edition)*. Springer, 2008.

[28] J. Zhu and B. Wang, “The optimal placement pattern for confident information coverage in wireless sensor networks,” *To Appear in IEEE Transactions on Mobile Computing (DOI: 10.1109/TMC.2015.2444397)*, 2015.