Arbitrary Obstacles Constrained Full Coverage in Wireless Sensor Networks*

Haisheng Tan[1], Yuexuan Wang[2], Xiaohong Hao[2], Qiang-Sheng Hua[1], and Francis C.M. Lau[1]

[1] Department of Computer Science, The University of Hong Kong, Pokfulam, Hong Kong, China
[2] Institute for Theoretical Computer Science, Tsinghua University, Beijing, 100084, China

Abstract. Coverage is critical for wireless sensor networks to monitor a region of interest and to provide a good quality of service. In many application scenarios, full coverage is required, which means every point inside the region (excluding the obstacles) must be covered by at least one sensor. The problem of using the minimum number of sensors to achieve full coverage for an arbitrary region with obstacles is NP-hard. Most existing coverage methods, such as contour-based ones, simply place sensors along the boundaries to cover the holes that are near the obstacles and the region boundary. These methods are inefficient especially when the obstacles or the region are irregular. In this paper, based on computational geometry, we design a full coverage method, which accurately finds the uncovered holes and places sensors efficiently for both the regular and irregular obstacles and regions. Specifically, we show that the more irregular the obstacles and the region are, the more sensors our method can save.

Keywords: Wireless sensor networks, Coverage, Obstacles, Computational Geometry.

1 Introduction

Wireless sensor networks (WSNs) have been applied extensively in military and civilian applications, and health care, for such purposes as environmental monitoring, intrusion detection, cancer monitoring, and smart agriculture [1]. Coverage is one of the fundamental issues in WSNs. It is measured by how well the region can be monitored and certain services can be provided.

Paper [3] studies how to place disks to fully cover a plane. The authors prove that it is asymptotically optimal, in terms of the number of disks used, to place disks

* The work presented in this paper was supported in part by Hong Kong RGC-GRF grants (7136/07E and 714009E), the National Basic Research Program of China Grant Nos. 2007CB807900, 2007CB807901, the National Natural Science Foundation of China Grant Nos. 60604033,60553001, and the Hi-Tech research and Development Program of China Grant No. 2006AA10Z216.

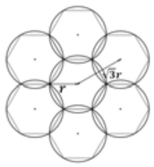

Fig. 1. The optimal placement pattern on a plane

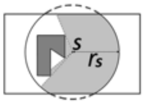

Fig. 2. Coverage of a sensor in a finite rectangle with an obstacle (the gray area is covered)

on the vertices of equilateral triangles (Figure 1). Several other optimal deployment patterns have also been proposed to achieve full coverage and k-connectivity ($k \leq 6$) on a plane in [4]. The Art Gallery problem [14] studies how to guard a gallery with the minimum number of cameras, where a location is guarded as long as there is line-of-sight from a camera. In this paper, in addition to the constraint of the limited sensing radius (r_s), the coverage of a sensor (s) is also affected by obstacles and the region boundary (Figure 2). The problem of using the minimum number of sensors to achieve full coverage for a region with obstacles has been proved to be NP-hard [5]. Paper [2] categorizes the approaches for coverage into three groups: force based [6,7], grid based [4,8,13], and computational geometry based [10,11,12]. The authors of [7] propose a virtual force algorithm (VFA) to enhance an initial random placement over a region with obstacles. Their method extremely depends on the sensors' mobility and energy. In [9], an efficient algorithm is designed for a robot to place sensors amid obstacles. Paper [12] deploys sensors with distance $\sqrt{2}r_s$ along the boundaries of obstacles and the region. Then Delaunay triangulation is applied to determine the positions for additional sensors for full coverage. The whole region is divided into single-row and multi-row regions in [13]; besides full coverage, the authors also guarantee network connectivity. In order to cover the holes, they deploy sensors along the boundaries of obstacles and the region with a constant distance based on a relationship between r_s and the sensor's communication radius r_c. To handle areas near obstacles and the region boundary, existing methods for stationary sensors [9,12,13] would simply place sensors along the boundaries with constant distances. We call these methods contour-based. To achieve full coverage, contour-based methods are effective for simple regular regions and obstacles, such as where the boundaries are long straight lines. However, for obstacles and regions that are arbitrarily irregular, such as when there are comb-shaped boundaries, these methods become inefficient. Without considering the specific shapes of boundaries, they may not produce full coverage unless additional sensors are placed at each turning point on the boundaries. In fact, even the best results by contour-based methods that place sensors with dynamic distances can be inefficient (Figure 3).

In this paper, in view of the inefficiency of existing deployment methods when handling irregular boundaries, we consider the shapes of boundaries explicitly. Using computational geometry techniques, our algorithms can find the holes accurately and cover them efficiently. Our method performs excellently for both regular and irregular obstacles and regions, including some extremely irregular

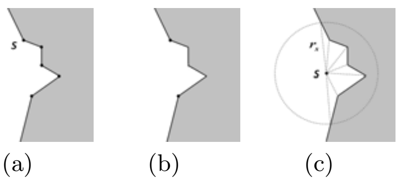

Fig. 3. Coverage near a part of the boundary: (a) the contour-based placement with a constant distance: 5 sensors, (b) the best for the contour-based methods: 2 sensors, (c) the optimal placement: 1 sensor

ones. The rest of the paper is organized as follows: in Section 2, we define the models and the problem; Section 3 describes our placement method; Section 4 gives the analysis and experimental results; Section 5 discusses some variations of our method; Section 6 concludes the paper and points out some future work.

2 Problem Definition

The region of interest, denoted as $\mathcal{A}$, is a 2D finite area with an arbitrary boundary and contains arbitrary obstacles. There are no isolated subregions, and the obstacles have no holes inside themselves. The area that needs to be covered is the region minus the obstacles. Both the region and obstacles are modeled as simple polygons of finite sizes on the 2D plane.

The sensors are stationary and homogenous with a fixed sensing radius r_s. Here we assume the binary sensor model: the sensing range of s is a disk centered at s with a radius of r_s; a point is covered by a sensor if it is within a distance of r_s and there is a line of sight from the sensor. The coverage of a sensor with obstacles, $Cov(s)$, may be a partial disk (Figure 2). If there are n obstacles inside $\mathcal{A}$, and the obstacle i occupies an area Obs_i, the total area occupied can be denoted as $\mathcal{O} = \cup_{i\in[1,n]} Obs_i$. And $Cov(s)$ is defined as:

$$Cov(s) = \{u \in \mathcal{A} - \mathcal{O} | \parallel u - s \parallel \leq r_s, ku + (1-k)s \in \mathcal{A} - \mathcal{O}, \forall k \in [0,1]\}$$

Full coverage for a sensing region $\mathcal{A}$ means every point in the area $\mathcal{A}-\mathcal{O}$ must be within $Cov(s)$ of at least one sensor s. Our sensor placement problem is defined as placing the minimum number of sensors in a finite region with arbitrary boundary and obstacles to achieve full coverage.

3 Sensor Placement Algorithms

To achieve full coverage for a region $\mathcal{A}$, our method works in four procedures: 1) to deploy a regular pattern over the region; 2) to find the uncovered holes; 3) to partition the holes into triangulations; and 4) to place sensors to cover the holes. In the following, we explain these procedures in details.

3.1 Optimal Regular Pattern Deployment (ORPD)

We firstly deploy the regular pattern in Figure 1, which is optimal for covering a plane, for the region $\mathcal{A}$ disregarding obstacles and the region boundary. The starting position can be randomly chosen at $(x, y) \in \mathcal{A}$. The sensors are placed at (x, y) as well as its six neighbors with distance $\sqrt{3}r_s$ in the pattern. Note that we assume a sensor can be 'placed' inside an obstacle or outside $\mathcal{A}$ in this procedure. For each sensor, as long as it is in $\mathcal{A}$, sensors are added at its neighbor locations. We continue doing this until every neighbor of the sensors inside $\mathcal{A}$ is occupied by one sensor. We use three lists $\mathcal{L}_{obs}$, $\mathcal{L}_{in}$ and $\mathcal{L}_{out}$ to keep track of the sensors placed inside obstacles $\mathcal{O}$, inside $\mathcal{A} - \mathcal{O}$, and outside $\mathcal{A}$ respectively. Figure 4 gives an example.

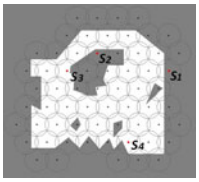

Fig. 4. The region after the ORPD: the white area is $\mathcal{A} - \mathcal{O}$

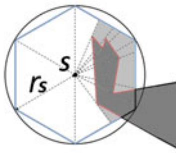

Fig. 5. Finding a hole (gray areas are *hole*(s))

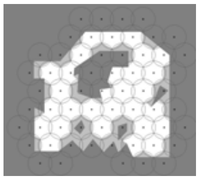

Fig. 6. Holes are found and merged (gray areas are holes)

3.2 Finding Holes

After the ORPD procedure, there may be some subareas near an obstacle or the region boundary, called *holes*, that are not covered, which is due to: 1) the lack of sensors: some sensors are placed into obstacles or outside the region, such as s_1 and s_2 in Figure 4; 2) blocking: some sensors' coverage is blocked by obstacles or the region boundary, such as s_3 and s_4. The arbitrary shapes of the obstacles and the region make it difficult to cover these holes efficiently. Our method overcomes this by firstly accurately finding the holes.

The ORPD procedure can be regarded as a tessellation of regular hexagons with edge r_s inscribed in the circles (Figure 1). Therefore, we model the sensing range of a sensor s by the regular hexagon, and denote it as $\mathcal{H}(s)$. Then, it is easy to find the holes caused by the lack of sensors. For each sensor deployed inside obstacles or outside $\mathcal{A}$, the hole of s, $hole(s)$ is the overlap, $\mathcal{H}_{in}(s)$, between $\mathcal{H}(s)$ and $\mathcal{A} - \mathcal{O}$. It is not as easy to calculate the holes caused by blocking (Figure 5). For a sensor deployed inside $\mathcal{A} - \mathcal{O}$, we also first calculate $\mathcal{H}_{in}(s)$. If $\mathcal{H}_{in}(s)$ is nonempty, holes may exist inside it. We denote the perimeter of an area as $P(area)$. $P(\mathcal{H}_{in}(s))$ consists of $P_H(s)$, the blue lines in Figure 5, and $P_A(s)$, the red lines excluding the intersection points with $P_H(s)$. Note that $P_H(s)$ is a part of $P(\mathcal{H}(s))$, and $P_A(s)$ is a part of $P(\mathcal{A} - \mathcal{O})$. Next, we draw directed lines from s to each vertex v of $\mathcal{H}_{in}(s)$ and extend them until reaching $P(\mathcal{H}(s))$. The line $\overrightarrow{sv}$ crosses $P(\mathcal{H}_{in}(s))$ at points $p_1, p_2, ..., p_n$ $(n > 0)$ listed in increasing distances from s. The segment of $\overrightarrow{sv}$ inside $\mathcal{H}(s)$ is denoted as $(s, p_1, p_2, ..., p_n)$. We set $p_0 = s$ and obtain Theorem 1 for finding the vertices of $hole(s)$.

Theorem 1. *Given* $(s, p_1, p_2, ..., p_n)$, *for* $\forall k \in [1, n)$, p_k *is a vertex of* $hole(s)$ *if and only if one of the following conditions is satisfied:*

1) $p_{k+1} \in P_H(s)$ *2)* $p_{k+1} \in P_A(s)$ *AND* $\frac{p_k+p_{k+1}}{2} \in \mathcal{H}_{in}(s)$
3) $p_{k-1} \in P_A(s)$ *AND* $\frac{p_k+p_{k-1}}{2} \in \mathcal{H}_{in}(s)$
4) $p_{k-1} \in P_A(s)$ *AND* $p_{k+1} \in P_A(s)$
For $k = n$, p_n *is a vertex of* $hole(s)$ *if and only if* $p_n \in P_H(s)$ *and* $p_{n-1} \neq s$.

Proof. As each $\overrightarrow{sv}$ stops at $P(\mathcal{H}(s))$, we can get $k = n \Longleftrightarrow p_k \in P(\mathcal{H}(s))$, and $k \in [1, n) \Longleftrightarrow p_k \in P_A(s)$. $P(\mathcal{H}(s)) - P_H(s)$ is inside obstacles or outside $\mathcal{A}$. Therefore, p_n is a vertex of $hole(s)$ if and only if $p_n \in P_H(s)$ and p_{n-1} is not s. As for $k \in [1, n)$, the theorem is proved by enumerating all the cases of the positions of p_k, p_{k-1} and p_{k+1} as follows:

p_k	p_{k-1}	p_{k+1}	p_k is a vertex of $hole(s)$
$P_A(s)$	s	$P_H(s)$	True
$P_A(s)$	s	$P(\mathcal{H}(s))-P_H(s)$	False
$P_A(s)$	s	$P_A(s)$	True iff $\frac{p_k+p_{k+1}}{2} \in \mathcal{H}_{in}(s)$
$P_A(s)$	$P_A(s)$	$P_H(s)$	True
$P_A(s)$	$P_A(s)$	$P(\mathcal{H}(s))-P_H(s)$	True iff $\frac{p_k+p_{k-1}}{2} \in \mathcal{H}_{in}(s)$
$P_A(s)$	$P_A(s)$	$P_A(s)$	True

$\square$

As the perimeter of $hole(s)$ can only consist of parts of $P(H_{in}(s))$ and segments on each directed line $\overrightarrow{sv}$, we have Theorem 2:

Theorem 2. *Connecting the vertices of* $hole(s)$ *along* $P(H_{in}(s))$ *and* $\overrightarrow{sv_i}, i = 1, 2, ..., m$, *where* m *is the number of vertices in* $H_{in}(s)$, *we get* $hole(s)$.

For a single sensor, $hole(s)$ may comprise a set of subholes. We merge the subholes that share at least a point. For all the sensors placed, we continue merging their holes, and denote the final set of holes as $\mathcal{HOLE}$. According to our finding process, we can see that the holes have good properties:

1) *narrow in width*: the width of the hole for a single s is no larger than r_s. After merging, each part of a hole in $\mathcal{HOLE}$ has a width no larger than $\sqrt{3}r_s$; otherwise there must be some sensors inside the hole, which is a contradiction;
2) *accurate in size*: if the sensing range of s is simulated as $\mathcal{H}(s)$ of area $\frac{3\sqrt{3}}{2}r_s^2$ as we do now, $\mathcal{HOLE}$ is exactly the uncovered area. Actually the range is a disk of area πr_s^2. Then $hole(s)$ is at most $(\pi - \frac{3\sqrt{3}}{2})r_s^2$ larger than the real uncovered area in the sensing range of s.

Algorithm 1 implements the whole procedure, and Figure 6 gives an example.

3.3 DT-Based Partition of Holes

To cover a hole in $\mathcal{HOLE}$, we partition it into triangles whose edges are no longer than r_s, so that a sensor can be placed at any of the three vertices of a triangle. We achieve this by the following three steps:

Step 1: Partitioning the long edges. If an edge of the hole is longer than r_s, $\lceil \frac{l}{r_s} \rceil - 1$ points are added to partition it evenly, such as $\overline{AB}$ in Figure 7. We treat the newly added points as vertices of the hole.

Algorithm 1. Finding Holes

```
Input:the region $\mathcal{A}$, obstacles $\mathcal{O}$, $\mathcal{L}_{in}, \mathcal{L}_{out}$ and $\mathcal{L}_{obs}$   Output: the holes $\mathcal{HOLE}$
 1: For each item $s \in \mathcal{L}_{out} \cup \mathcal{L}_{obs}$ do
 2:     $hole(s) = \mathcal{H}(s) \cap (\mathcal{A} - \mathcal{O})$
 3:     if $hole(s) \neq \varnothing$, $\mathcal{HOLE} = \mathcal{HOLE} \cup hole(s)$
 4: End For
 5: For each item $s \in \mathcal{L}_{in}$ do
 6:     $\mathcal{H}_{in}(s) = \mathcal{H}(s) \cap (\mathcal{A} - \mathcal{O})$
 7:     If $\mathcal{H}_{in}(s) \neq \varnothing$, Then
 8:         Finding the vertices of $hole(s)$ /* Theorem 1 */
            Compute and merge subholes to get $hole(s)$; update $\mathcal{HOLE}$ /* Theorem 2 */
 9:     End If
10: End For
11: For each pair of items $hole_1$ and $hole_2$ in $\mathcal{HOLE}$
12:     if $hole_1 \cap hole_2 \neq \varnothing$, $hole = hole_1 \cup hole_2$ and update $\mathcal{HOLE}$
13: End For
```

Step 2: Applying Delaunay triangulation. We apply DT over the vertices of the hole, and get a triangulation of the hole. Here DT is chosen because 1) it is efficient ($O(nlogn)$ time, where n is the number of vertices), and 2) the minimum angle of all the triangles is maximized thus avoiding skinny triangles [14]. Based on the advantage of DT and the property that our holes are narrow, the triangulation will add no or just a few long edges that need to be handled in next step.
Step 3: Partitioning triangles with an edge longer than r_s, such as $\overline{CD}$ in Figure 7. Also, to avoid skinny triangles, we always partition the longest edge among all the triangles. This step stops until all the edges are bounded by r_s and we finally get a partition, $\mathcal{T}_{hole}$, over the hole.

The partition for a hole from Figure 6 is illustrated in Figure 7. After handling all the holes in $\mathcal{HOLE}$, we get a set of partitions, $\mathcal{T}_{holes}$, as described in Algorithm 2.

3.4 Placing Sensors to Cover Holes

Based on the partition, full coverage is equivalent to satisfying the requirement that at least one vertex of each triangle in $\mathcal{T}_{holes}$ is placed a sensor. For each $\mathcal{T}_{hole}$ in $\mathcal{T}_{holes}$, we first compute its dual graph, $\mathcal{D}(\mathcal{T}_{hole})$, which has a node v for every triangle $t(v)$ in $\mathcal{T}_{hole}$. Two nodes u and v form an edge if $t(v)$ and $t(u)$ share at least a point. If $\mathcal{D}(\mathcal{T}_{hole})$ is a tree, it is 3-colorable. When performing DFS (Depth-First Search) on the tree, for each v being visited except the root, there must be one or two (when $t(v)$ shares only a vertex with the triangle visited in the last step) free nodes uncolored in $t(v)$. However, in our case, $\mathcal{D}(\mathcal{T}_{hole})$ may be a graph with cycles and not 3-colorable. In order to ensure that every triangle has three different colored vertices, we allow a vertex to be ‘doubly colored’. When performing DFS on the graph, for the last node v in a cycle, the free vertex in $t(v)$ has been colored. We check whether the three vertices of $t(v)$ have had three different colors. If yes, we go on to visit the next node directly; if no, we append the absent color to a colored free vertex and make it doubly colored,

Algorithm 2. DT-based partition of Holes

```
Input: the set of holes $\mathcal{HOLE}$    Output: a partition $\mathcal{T}_{holes}$ over $\mathcal{HOLE}$
 1: For each hole in $\mathcal{HOLE}$ do
 2:    Initialize an array $\mathcal{L}_p$, a partition $\mathcal{T}_{hole}$ and a sorted array $\mathcal{L}_e$ empty
 3:    $\mathcal{L}_p = \mathcal{L}_p \cup$ vertices of hole
 4:    For each edge $e$ of hole do          /* Step 1 */
          add $\lceil \frac{length(e)}{r_s} \rceil - 1$ points evenly on $e$ and append the points to $\mathcal{L}_p$
       End For
 5:    Do DT over $\mathcal{L}_p$ and store the triangulation of hole to $\mathcal{T}_{hole}$      /* Step 2 */
 6:    Insert edges in $\mathcal{T}_{hole}$ longer than $r_s$ to $\mathcal{L}_e$ in decreasing order of lengths
 7:    While($\mathcal{L}_e$ is nonempty) do          /* Step 3 */
 8:       add $\lceil \frac{length(e)}{r_s} \rceil - 1$ points evenly on the first item $e$ and remove it
 9:       For each triangle containing $e$ do
             connect the points added to the vertex that is not on $e$; update $\mathcal{T}_{hole}$
10:          if there are edges added longer than $r_s$, insert them to $\mathcal{L}_e$
11:       End For
12:    End While
13:    $\mathcal{T}_{holes} = \mathcal{T}_{hole} \cup \mathcal{T}_{holes}$
14: End For
```

such as the vertex D in Figure 7. After coloring, we choose the smallest group of the vertices with the same color to place our sensors. Because for each cycle in $\mathcal{D}(\mathcal{T}_{hole})$, there is at most one vertex in $\mathcal{T}_{hole}$ doubly colored, we have Theorem 3:

Theorem 3. *For a hole h, if it has n vertices in $\mathcal{T}_h$ and c cycles in $\mathcal{D}(\mathcal{T}_h)$, $\lfloor \frac{n+c}{3} \rfloor$ sensors can always fully cover it.*

After placing sensors to cover all the holes, together with the sensors in $\mathcal{L}_{in}$, we can get full coverage of the region $\mathcal{A}$ (Figure 8).

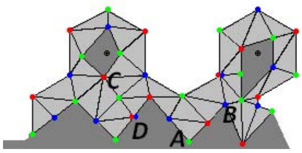

Fig. 7. Partition a hole into triangles and color the vertices.

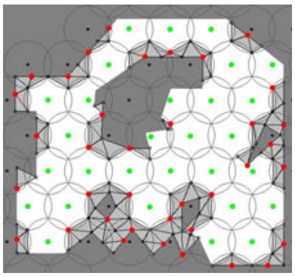

Fig. 8. Full coverage: red and green points are sensors

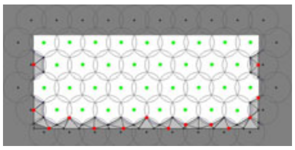

Fig. 9. Cover a rectangle of long edges: red and green points are sensors

4 Analysis and Experiments

Because of the arbitrary obstacles and the region, to analyze the bounds on the number of sensors for full coverage, we need to use parameters from both the input region and our placement algorithms. For the region $\mathcal{A}$, n_v is the number of vertices on $P(\mathcal{A} - \mathcal{O})$, and the length of each edge i on $P(\mathcal{A} - \mathcal{O})$ is len_i. For the holes in $\mathcal{HOLE}$, the number of vertices located on $\{P_H(s)|hole(s) \neq \varnothing\}$ is n_{vph}, and the number of vertices on $\{P_A(s)|hole(s) \neq \varnothing\}$ is n_{vpa}. The number of sensors to cover $\mathcal{HOLE}$ is n_{ch}. k is the number of vertices to divide long edges after applying DT over $\mathcal{HOLE}$. Recall that there are no or at most a few edges longer than r_s after DT, so k is small. m is the total number of cycles in the dual graphs of $\mathcal{T}_{holes}$. m is also small and closely related to the shapes and positions of obstacles and the region boundary. The size of $\mathcal{L}_{in}$ is n_{in}, and the number of vertices in $\mathcal{T}_{holes}$ is n_{vt}. Then the number of sensors to achieve full coverage of $\mathcal{A}$ is:

$$N = n_{in} + n_{ch} \leq n_{in} + \lfloor (n_{vt} + m)/3 \rfloor \quad /* Theorem\ 3 */ \tag{1}$$

$$= n_{in} + \lfloor \frac{n_{vph} + n_{vpa} + \sum_i (\lceil \frac{len_i}{r_s} \rceil - 1) + k + m}{3} \rfloor \tag{2}$$

$$\leq n_{in} + \lfloor \frac{n_{vph} + n_v + \sum_i (\lceil \frac{len_i}{r_s} \rceil - 1) + k + m}{3} \rfloor \quad /* n_{vpa} \leq n_v */ \tag{3}$$

$$N \geq \frac{area(\mathcal{A} - \mathcal{O})}{area(\mathcal{H}(s))} = \frac{area(\mathcal{A} - \mathcal{O})}{3\sqrt{3}r_s^2/2} \tag{4}$$

n_{in} is the asymptotically optimal number of sensors to cover the region excluding $\mathcal{HOLE}$ by the ORPD procedure. n_{vph} approximates to $n_{vpa} + \sum_i (\lceil \frac{len_i}{r_s} \rceil - 1)$ in most cases, and for irregular obstacles and regions, n_{vph} can be much smaller. So, the upper bound of the sensors for holes is nearly $\frac{2}{3}(n_v + \sum_i (\lceil \frac{len_i}{r_s} \rceil - 1))$. However, for contour-based methods, the sensors placed along boundaries can reach $n_v + \sum_i (\lceil \frac{len_i}{r} \rceil - 1)$, where r is the constant placement distance not larger than $\sqrt{2}r_s$.

We carried out experiments to validate the above analysis and to show the effectiveness of our approach for both regular and irregular cases. The example in Figure 8 illustrates the effectiveness of our approach in handling comb-shaped boundaries. Figure 9 is to cover a rectangle, which illustrates the handling of long straight lines. Moreover, we conducted simulations for four types of regions (Figure 10) using three types of sensors with different r_s. Figure 11 compares our results with the contour-based ones'. 'Contour-based-1' is to first place sensors along the boundaries, and then add extra sensors for full coverage via a near-optimal placement. 'Contour-based-2' applies the ORPD procedure and then places sensors along the boundaries to cover the holes. When executing ORPD in 'Contour-based-2' and in our method, we try different locations for the first sensor, and choose the one leading to the fewest sensors for full coverage. Greedily, we place the first sensor $\frac{r_s}{2}$ away from the longest boundary and rotate the regular pattern to fully cover the longest boundary. We can see that the more irregular the obstacles and region are, the more sensors our method can save.

5 Discussion

We have considered only *opaque obstacles* which neither allow sensors to be deployed inside nor allow the sensing signal to pass through. There exist other

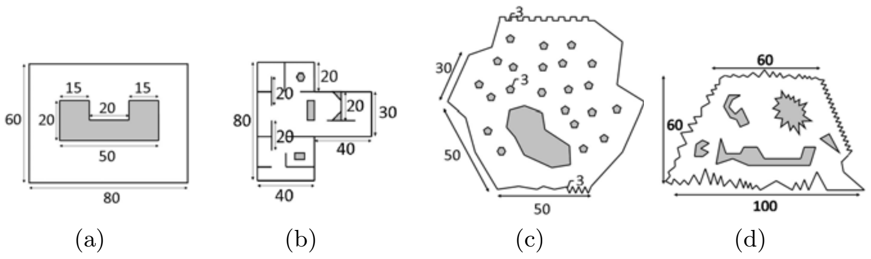

Fig. 10. Four types of regions: (a) a regular region and obstacles (b) an indoor-like region (c) an outdoor-like region (d) an extremely irregular region and obstacles

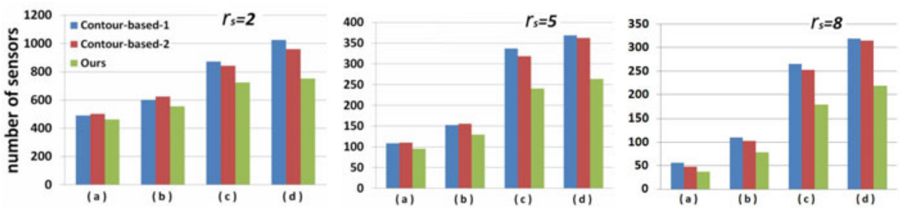

Fig. 11. Comparison of the contour-based methods with ours in above 4 regions

kinds of obstacles such as *transparent obstacles* (ponds are examples) which do not allow sensors placed inside but allow signals to pass through. When the obstacles and region boundary are all transparent, the holes appearing after the ORPD procedure are only those due to the lack of sensors. We can compute $\mathcal{H}_{in}(s)$ for each $s \in \mathcal{L}_{out} \cup \mathcal{L}_{ob}$. For each nonempty $\mathcal{H}_{in}(s)$, we need to add at most 5 sensors to cover it [5]. Setting $n = |\{s \in \mathcal{L}_{out} \cup \mathcal{L}_{ob} | \mathcal{H}_{in}(s) \neq \varnothing\}|$, we have the upper bound of the sensors needed as $n_{in} + 5n$. Note that the upper bound in [5] is incorrect because it sums up each $\mathcal{H}_{in}(s)$ without considering their positions which are as crucial as their sizes. An example to refute their upper bound is a $w \times l$ rectangle with $w \to 0$.

In practice, due to the inherent uncertainty of a sensor's sensing ability, *the probabilistic sensor model* is more reasonable than the binary model [7]. As full coverage is achieved in our placement, for any point in $\mathcal{A} - \mathcal{O}$, there must be at least one sensor s within a distance r_s. Therefore, our coverage probability for any point is not smaller than $e^{-\lambda r_e^\beta}$, where λ, r_e and β are parameters in the model. If a higher probability is required, we need to use a virtual sensing radius r instead of r_s, where $r_s - r_e \le r < r_s$, so that the probability $e^{-\lambda(r+r_e-r_s)^\beta}$ is guaranteed.

For arbitrary obstacles, the effective coverage of a sensor can be infinitesimal. The number, shapes, sizes as well as relative positions of obstacles make it hard to design and analyze algorithms for full coverage. In practice, obstacles are rarely extremely irregular. Therefore, we may choose to study coverage for a limited number of special sizes and shapes of obstacles. Moreover, instead of full coverage, we may also choose to ignore holes smaller than a certain predefined threshold.

6 Conclusion and Future work

In this paper, we propose an efficient full coverage method for a sensing region with arbitrary boundary and obstacles. We achieve this by carefully studying how the obstacles and the region boundary block coverage. The holes arising from an optimal regular pattern deployment are efficiently and accurately calculated. Algorithms based on Delaunay triangulation and 3-coloring are designed to cover these holes. Compared with other methods, the proposed method can achieve full coverage while dramatically saving sensors especially when the region and obstacles are irregular. Besides the extension mentioned in the previous section, future work can be carried out in many directions, such as multiple connectivity and coverage with obstacles, 3D coverage and the surface coverage [15] with obstacles, coverage by mobile and heterogeneous sensors, etc.

References

1. Wu, J.: Handbook on Theoretical & Algorithmic Aspects of Sensor. In: Ad Hoc Wireless, and Peer-to-Peer Networks. Auerbach Publication, US (2006)
2. Aziz, N.A.A., Aziz, K.A., Ismail, W.Z.W.: Coverage Strategies for Wireless Sensor Networks. World Academy of Science, Engineering and Technology 50 (2009)
3. Kershner, R.: The Number of Circles Covering a Set. American Journal of Mathematics 61, 665–671 (1939)
4. Bai, X., Xuan, D., Yun, Z., Lai, T.H., Jia, W.: Complete Optimal Deployment Patterns for Full-Coverage and k-Connectivity ($k \leq 6$) Wireless Sensor Networks. In: Proc. of ACM MobiHoc 2008 (2008)
5. Shyam, M., Kumar, A.: Obstacle Constrained Total Area Coverage in Wireless Sensor Networks. CoRR abs/1001.4753 (2010)
6. Howard, A., Poduri, S.: Potential Field Methods for Mobile-Sensor- Network Deployment. In: Bulusu, N., Jha, S. (eds.) Wireless Sensor Networks A System Perspective. Artech House, London (2005)
7. Zou, Y., Chakrabarty, K.: Sensor deployment and target localization based on virtual forces. In: IEEE INFOCOM 2003 (2003)
8. Shen, X., Chen, J., Wang, Z., Sun, Y.: Grid Scan: A Simple and Effective Approach for Coverage Issue in Wireless Sensor Networks. IEEE International Communications Conference, volume: 8, pp. 3480-3484. (2006)
9. Chang, C.Y., Chang, C.T., Chen, Y.C.: Obstacle-Resistant Deployment Algorithms for Wireless Sensor Networks. IEEE Trans. on Veh. Tech. (2009)
10. Wang, G., Cao, G., Porta, T.L.: Movement-Assisted Sensor Deployment. In: IEEE INFOCOM 2004, vol. 4, pp. 2469–2479 (2004)
11. Megerian, S., Koushanfar, F., Potkonjak, M., Srivastava, M.: Worst and Best-Case Coverage in Sensor Networks. IEEE Trans. on Mob. Comput. 4(1), 84–92 (2005)
12. Wu, C.H., Lee, K.C., Chung, Y.C.: A Delaunay Triangulation based method for wireless sensor network deployment. Computer Communications 30 (2007)
13. Wang, Y.C., Hu, C.C., Tseng, Y.C.: Efficient Placement and Dispatch of Sensors in a Wireless Sensor Network. IEEE Trans. on Mob. Comput. 7(2) (2008)
14. Berg, M., Cheong, O., Kreveld, M.V., Overmars, M.: Computational geometry: algorithms and applications, 3rd edn. Springer Press, Heidelberg (2008)
15. Zhao, M.C., Lei, J., Wu, M.Y., Liu, Y., Shu, W.: Surface Coverage in Wireless Sensor Networks. In: IEEE INFOCOM 2009, pp. 109–117 (2009)