Obstacle-Resistant Deployment Algorithms for Wireless Sensor Networks

Chih-Yung Chang, *Associate Member, IEEE*, Chao-Tsun Chang, Yu-Chieh Chen, and Hsu-Ruey Chang

***Abstract*—Node deployment is an important issue in wireless sensor networks (WSNs). Sensor nodes should be efficiently deployed in a predetermined region in a low-cost and high-coverage-quality manner. Random deployment is the simplest way to deploy sensor nodes but may cause unbalanced deployment and, therefore, increase hardware costs and create coverage holes. This paper presents the efficient obstacle-resistant robot deployment (ORRD) algorithm, which involves the design of a node placement policy, a serpentine movement policy, obstacle-handling rules, and boundary rules. By applying the proposed ORRD, the robot rapidly deploys a near-minimal number of sensor nodes to achieve full sensing coverage, even though there exist unpredicted obstacles with regular or irregular shapes. Performance results reveal that ORRD outperforms the existing robot deployment mechanism in terms of power conservation and obstacle resistance and, therefore, achieves better deployment performance.**

***Index Terms*—Deployment, obstacles, wireless sensor network (WSNs).**

I. Introduction

Wireless sensor networks (WSNs) are composed of many sensor nodes that are embedded with simple processors, little memory, tiny sensing materials, and energy-limited batteries [1]. Those tiny and low-cost sensor nodes are deployed in large quantities in a specific sensing region for application to environment monitoring or military detection. One of the most important functions of the sensor node is to collect pertinent information such as earthquake intensity, light, or temperature measurement in environmental monitoring applications or enemy or chemical gas detection in military detection applications [2]–[4]. However, the accuracy of the information depends on the quality of coverage within the sensing region.

In the literature, existing deployment algorithms can be classified into the following three categories: 1) stationary sensor [8], [9]; 2) mobile sensor [10]–[13]; and 3) mobile robot [14]–[18]. Several random deployment schemes [8], [9] have been proposed for the deployment of stationary sensors. Random deployment is simple and easy to implement. In a randomly deployed WSN, a number of coverage maintenance protocols have been proposed [5]–[7]. However, the number of deployed sensors is much larger than what is actually required to ensure full coverage. Randomly deploying stationary sensors may result to an inefficient WSN, where some areas are densely deployed while the other areas have a low density deployment. The denser deployment in some areas increases the hardware cost, whereas a sparse deployment in the other areas results to coverage holes or network partitions. As a result, there is no guarantee for full coverage, and a considerable hardware costs are needed.

Some other studies [10], [11] address the coverage problems in a mobile WSN, which is composed of a large number of stationary sensors and a few mobile sensors. After an initial phase of random deployment of stationary sensors, mobile sensors are coordinated to compute for their target locations according to the information regarding the holes in the monitoring area and then move to the target locations to heal the existing coverage holes. However, hardware costs cannot be reduced in those areas that were densely deployed with stationary sensors. Previous works [12], [13] further considered the mobile WSN, which is wholly composed of mobile sensors. With mobility support, the neighboring mobile sensors can cooperatively adjust their locations to achieve full coverage. However, each mobile sensor requires additional hardware cost that supports the mobility, and considerable energy consumption is required for each mobile sensor that moves from one location to another.

An alternative [14]–[18] is the use of robots in deploying static sensors in a given region. The robot explores the environment and deploys a stationary sensor on the target location from time to time. Robot deployment can achieve full coverage with fewer sensors and increase the sensing effectiveness of stationary sensors to guarantee full coverage and connectivity. In addition, the robot may perform other missions such as hole detection, redeployment, monitoring, and so on. However, the unpredicted obstacles can challenge robot deployment and make a great impact on the deployment efficiency. It requires much more effort to develop a robot deployment mechanism that uses fewer sensors for full coverage and power efficiency, even though the monitored regions contain unpredicted obstacles.

In [15], the robot deploys the sensors according to the predefined direction priorities of south, west, north, and east. Each sensor counts the time interval that the robot does not explore for each direction. Deployed sensors within the communication range of the robot may guide the robot's movement by

Manuscript received July 3, 2007; revised December 21, 2007, May 16, 2008, and October 9, 2008. First published December 9, 2008; current version published May 29, 2009. The review of this paper was coordinated by Prof. A. Boukerche.

C.-Y. Chang, Y.-C. Chen, and H.-R. Chang are with the Department of Computer Science and Information Engineering, Tamkang University, Tamsui 25137, Taiwan (e-mail: cychang@mail.tku.edu.tw; ycchen@wireless.cs.tku.edu.tw; hrchang@wireless.cs.tku.edu.tw).

C.-T. Chang is with the Department of Information Management, Hsiuping Institute of Technology, Taichung 41280, Taiwan (e-mail: cctas@mail.hit.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2008.2010619

TABLE I
COMPARISON OF THE MAIN CHARACTERISTICS OF THE PROPOSED SCHEME WITH PREVIOUS SCHEMES

Studies	Static Sensor	Mobile Sensor	Mobile Robot	Deployed Density	Full Coverage	# of Sensors	Boundary/ Obstacle	Obstacle Information
[8-9]	Used	No	No	Nonuniform	No	More	No/No	N/A
[10-11]	Used	Used	No	Nonuniform	Yes	Medium	No/No	N/A
[12-13]	No	Used	No	Uniform	Yes	Fewer	No/No	N/A
[14, 16]	Used	No	Used	Uniform	Yes	Fewer	No/No	N/A
[15, 18]	Used	No	Used	Uniform	No	Fewer	All Regular	Unknown
[17]	Used	No	Used	Uniform	Yes	Fewer	All Regular	Known
ORRD	Used	No	Used	Uniform	Yes	Fewer	All Irregular	Unknown

suggesting a suitable direction with the maximum time interval to the robot. When the robot receives the different suggestions, it integrates them and selects the best direction for patrol and/or sensor deployment. To reduce the impact of obstacles, previous work [18] proposes four traveling orders, namely, 1) random, 2) cross, 3) line, and 4) circle, as the movement options of the robot. Since each subsequent movement is determined by the predefined rules regardless of the obstacles relative to the robot, both approaches of Batalin and Sukhatme [15], [18] cannot guarantee full coverage and may even introduce several sensing redundancies when the robot encounters the obstacles. In addition, there is no discussion about how to handle the irregular obstacles. To handle the obstacles problem, Wang *et al.* [17] proposed a centralized algorithm that uses global obstacle information to calculate the best deployment location of each sensor. Although the proposed mechanism achieves full coverage and connectivity using fewer stationary sensors, global obstacle information is required. However, since global information is not possible in an unexplored region, the developed mechanism could only be used for limited applications.

This paper presents the efficient obstacle-resistant robot deployment (ORRD) algorithm, which involves the design of a node placement policy, a serpentine movement policy, and obstacle-handling rules. Performance analysis of the proposed ORRD is given in terms of the number of required sensors for an environment containing several irregular obstacles. Simulation results revealed that the proposed ORRD overcomes the unpredicted obstacles and deploys fewer static sensors but achieves higher coverage percentages compared to existing deployment algorithms. Table I summarizes the characteristics of the related studies and the proposed ORRD. In comparison, the major difference between this paper and related studies is that the obstacles are unknown and irregular, which raises challenges in designing an efficient robot deployment mechanism.

The rest of this paper is organized as follows. Section II introduces the environment and basic concepts of this paper. The proposed ORRD algorithm is presented in Section III, while Section IV proposes the boundary problems and its handling rules. Section V analyzes the performance efficiency of ORRD. Section VI investigates the performance of ORRD by simulation study, while Section VII concludes this work and gives some suggestions for future studies.

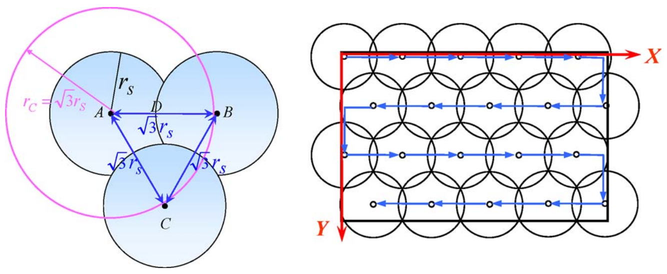


Fig. 1. (a) Optimal deployment of the three nearby sensors A, B, and C. (b) Serpentine movement deployment.

II. NETWORK ENVIRONMENT AND BASIC CONCEPTS

A. Network Environment

Consider a single robot with limited static sensor nodes that is equipped with a compass that keeps track of the direction of its movements. Assume that the boundaries of a given monitoring region are known by the robot. Further, assume that the robot is initially located at the upper left corner of the monitoring region. This constraint can be removed when the obstacle is taken into account. Let r_c and r_s denote the communication and sensing ranges, respectively. Herein, we assume that r_c is larger than $\sqrt{3}r_s$.

B. Basic Concepts

An optimal robot deployment means that the robot deploys a minimal number of sensors to achieve the purpose of full coverage. To achieve an optimal deployment, the overlapping region of neighboring sensor nodes should be strictly controlled [17]. Fig. 1(a) illustrates the basic requirement for optimal deployment. Let nodes A, B, and C be the three neighboring sensors. The optimal deployment can be reached if the three sensor nodes intersect with each other at one point. In this situation, the distance of any pair of A, B, and C is exactly equal to $\sqrt{3}r_s$. Based on this observation, a *deployment policy* directs the robot to deploy a sensor every $\sqrt{3}r_s$.

In addition to the *deployment policy*, a serpentine movement policy is employed to guide the robot's movement. Fig. 1(b) depicts the serpentine movement when the robot deploys a sensor every distance interval of $\sqrt{3}r_s$.

Furthermore, the proposed ORRD aims to efficiently overcome the unpredicted obstacle and develops obstacle-handling

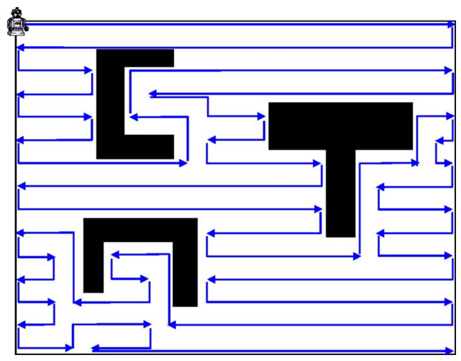


Fig. 2. Deployment by applying the obstacle-handling mechanism.

rules to alleviate the negative impact of obstacles on the deployment task. As shown in Fig. 2, the black blocks represent three obstacles with different shapes, and the directional lines denote the trajectory of the robot's movement by applying the proposed ORRD. To highlight the movement trajectory, sensors deployed in the WSN are not shown in the figure. The trajectory shows that ORRD takes into consideration the unpredicted obstacles and achieves full coverage even though obstacles exist in the monitoring region.

III. ORRD Mechanism

A. *Legal Movement Pattern*

This section presents the serpentine deployment mechanism that enables the robot to deploy the minimum number of sensor nodes to achieve full coverage and overcome unpredicted obstacles. The following paragraph will introduce the basic rules for simple serpentine deployment without considering any obstacle. Afterward, the obstacle-resistant deployment rules are proposed.

The robot will deploy a sensor node after each movement for a distance of $\sqrt{3}r_s$. To achieve the optimal deployment, the movement of the robot should be one of the six legal patterns as shown in Fig. 3(a). The six types of basic movement are referred to as the *legal patterns for basic movement*. Types 1 and 2 are used when the robot moves toward the east and west directions, respectively. As shown in Fig. 1(a), the sensor nodes deployed on the ith row are located on the perpendicular bisector of two neighboring sensors deployed on the $(i-1)$th row. When the robot encounters a boundary or obstacle, it should deploy the sensor on the next row. That is, the robot should move toward the south. Type 3 will be used when the robot moves toward the west direction but encounters the left boundary or obstacle. In this case, the robot will initially move toward the south for a distance of $(3/2)r_s$ and then moves toward the east for a distance of $(\sqrt{3}/2)r_s$. Similarly, type 4 is used when the robot moves toward the east but encounters the right boundary or obstacle. To overcome the unpredicted obstacles, sometimes, the robot would move toward the north direction. Types 5 and 6 are used when the robot tries to overcome the obstacle and moves toward the north direction. Fig. 3(b) shows an example of the deployment by applying the six types of basic movement.

B. *Simple Serpentine Robot Deployment*

In the serpentine movement, the robot stays in either the East or West states. The East and West states denote that the robot is currently moving toward the east and west directions, respectively. Each state has two legal patterns of basic movement, and the robot chooses one pattern as its movement policy according to the priority. Table II depicts the two legal patterns of basic movement for each state and their priorities. In both the East and West states, Prefer Direction 1 has a higher priority that enables the robot moving along the east and west directions, respectively. Prefer Direction 2 has a lower priority and will be applied in case the movement in Prefer Direction 1 is a failure. Prefer Direction 2 will guide the robot moving toward the south. As soon as Prefer Direction 2 is applied, the robot's state should be changed.

For simplicity, we assume that the robot is initially located at the upper left corner of the monitoring region. The initial state of the robot will be the East state. According to Table II, the robot will determine the direction of the next movement according to Prefer Direction 1. Hence, it moves toward the east for a distance of $\sqrt{3}r_s$ and then deploys a sensor. The robot will continuously move toward the east direction and deploys a sensor after each basic movement until it encounters the right boundary. Since the right boundary may cause the east movement to fail, the robot makes a decision according to Prefer Direction 2. Hence, it moves toward the south direction for a distance of $(3/2)r_s$ and then moves toward the west direction for a distance of $(\sqrt{3}/2)r_s$. As soon as the robot changes its movement direction, the state of the robot is also changed from East to West. After that, the robot makes a decision according to Prefer Direction 1 and, hence, moves toward the west direction until the left boundary is encountered. Note that the robot deploys a sensor node at a distance of every $\sqrt{3}r_s$. The East and West states can take place by turns in the serpentine movement, as shown in Fig. 1(b).

C. *Obstacle-Resistant Serpentine Robot Deployment*

1) Impact of Obstacles: This subsection further considers the existence of obstacles and develops an obstacle-resistant serpentine deployment mechanism. Fig. 4 depicts two main challenges for the developed simple serpentine movement scheme. As shown in Fig. 4, the black region represents an obstacle, and the directional line represents the trajectory of the robot's movement by applying the simple serpentine deployment mechanism proposed in Section III-B. Consequently, the deployment results to several sensing holes. The sensing holes A and B exist due to the obstacle, while the sensing holes C and D exist because the robot encountered the boundary of the obstacle and the monitoring region, respectively. This section proposes an obstacle-resistant serpentine deployment algorithm to overcome both the obstacle and the boundary problems. The deployment algorithm shown later is divided into the obstacle-handling rules and the boundary handling rules. The obstacle-handling rules enable the robot to efficiently overcome the obstacle and deploy sensor nodes in the monitoring region, while the boundary handling rules address the boundary problem.

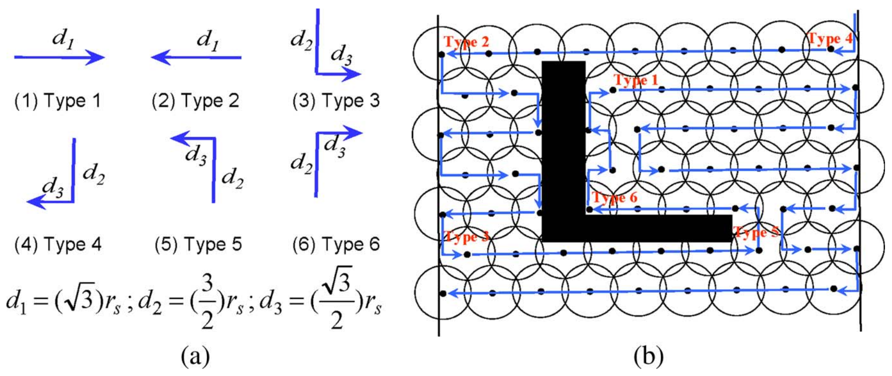


Fig. 3. Six types of basic movements and their usages. (a) Six legal patterns for basic movement. (b) Scenario of applying six types of basic movement to overcome obstacles.

TABLE II
SIMPLE SERPENTINE DEPLOYMENT

States	Prefer Direction 1	Prefer Direction 2
East	→(Type 1)	↲ (Type 4)
West	←(Type 2)	↳ (Type 3)

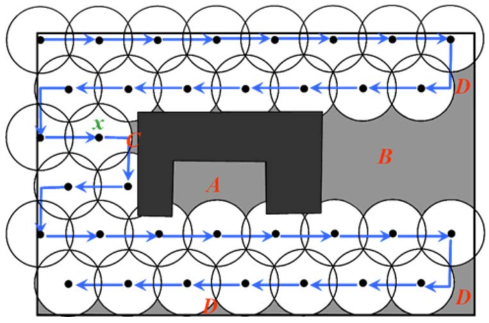


Fig. 4. Applying simple serpentine deployment results several holes due to the existence of obstacles and boundary.

TABLE III
CHECK DIRECTIONS FOR OVERCOMING THE OBSTACLE

States	Check Direction 1	Check Direction 2	
East	←(Type 2)	↰ (Type 5)	↱ (Type 6)
West	→(Type 1)	↱ (Type 6)	↰ (Type 5)

TABLE IV
OBSTACLE-RESISTANT SERPENTINE MOVEMENT RULE

States	Check Direction 1	Check Direction 2		Prefer Direction 1	Prefer Direction 2	
East	←	↰	↱	→	↳	↲
West	→	↱	↰	←	↲	↳

2) Obstacle-Handling Rules: Assume that the robot stays in the East state. The robot repeatedly applies movement type 1, and hence, it moves to the east direction. As the robot encounters the right boundary, it checks Prefer Direction 2 according to the simple serpentine movement mechanism and applies movement type 4. Thus, the robot moves to the south direction for a distance of $(3/2)r_s$ and then moves to the west direction for a distance of $(\sqrt{3}/2)r_s$. However, by moving this way, the robot has no opportunity for the robot to visit the north and west directions to redeploy sensors in the existing hole. Hence, before the robot applies the simple serpentine movement scheme, it should first check if a sensing hole exists in the north or west direction. This implies that the movement to the north and west directions should be prior to the current Prefer Direction 1.

Table III lists the *check directions* for the robot to further check if any sensing hole existed in the check direction. Before moving according to Prefer Direction 1, the robot will check the check direction first prior to its movement. In case the sensors deployed in the check direction exist, the robot will apply the simple serpentine movement scheme and utilize Table II to decide the next movement direction.

There are two check directions for each state. In case the robot stays in the East state, it initially checks *Check Direction 1*. If there is no sensor deployed in the west direction, the next movement direction will be to the west. Otherwise, the robot will check *Check Direction 2*. If the north direction did not deploy any sensor, the robot will move to the north direction in the next movement. If, fortunately, a hole does not exist in the two check directions, the robot will further apply the simple serpentine movement scheme that utilizes Table II to determine the next movement direction. If the robot stays in the West state, the situation becomes similar to the East state and is, hence, omitted herein.

Table IV summarizes the check directions and prefer directions with priorities. As shown in Table IV, the robot should select one of the six movement types according to their priorities. In general, if the robot stays in the East state, the six movement types have the following priorities:

$$\underset{(\leftarrow)}{\text{Type 2}} > \underset{(↰)}{\text{Type 5}} > \underset{(↱)}{\text{Type 6}} > \underset{(\rightarrow)}{\text{Type 1}} > \underset{(↳)}{\text{Type 3}} > \underset{(↲)}{\text{Type 4}}.$$

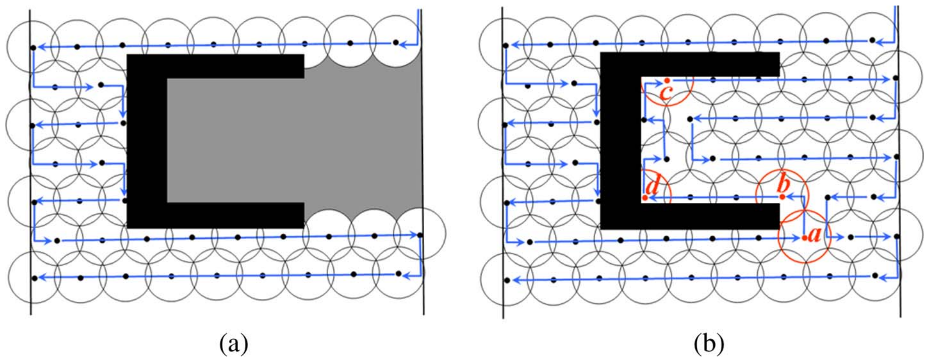


Fig. 5. (a) Applying the simple serpentine deployment scheme results an inner-concave sensing hole. (b) Proposed obstacle-resistant serpentine deployment mechanism can overcome the obstacle and achieve full coverage.

Otherwise, the six movement types have the following priorities:

$$\underset{(\rightarrow)}{\text{Type 1}} > \underset{(\ulcorner\!\rightarrow)}{\text{Type 6}} > \underset{(\leftarrow\!\urcorner)}{\text{Type 5}} > \underset{(\leftarrow)}{\text{Type 2}} > \underset{(\leftarrow\!\lrcorner)}{\text{Type 4}} > \underset{(\llcorner\!\rightarrow)}{\text{Type 3}}.$$

Note that if the robot fails to move to a certain direction with a higher priority, the robot will try the next higher priority until a successful movement occurs. The following four rules summarize the aforementioned obstacle-handling algorithm. An attempt to a movement direction is said to be a failure if there exists a deployed sensor, obstacle, or boundary in that direction.

Rules 1 and 2 are mainly designed for handling obstacles, while rules 3 and 4 are designed for the simple serpentine movement. In each movement, the robot initially checks the rules from rule 1 to rule 4 in order and then executes one of the four rules to determine the direction of the next movement. The robot moves toward the selected direction for a predefined distance, as shown in Fig. 5(a), and then deploys a static sensor.

/∗ ***Rules 1 and 2 are designed for overcoming obstacle.*** ∗/

Rule 1: The robot checks Check Direction 1 (which is in the opposite direction to Prefer Direction 1) for a possible movement. If the attempt to this direction is a failure, then the robot executes the next rule. Otherwise, the robot will move toward Check Direction 1 for a distance of $\sqrt{3}r_s$.

Rule 2: The robot tries to move to Check Direction 2. If the attempt to this direction is a failure, then the robot checks Rule 3 subsequently. Otherwise, the robot will move toward Check Direction 2 for a distance of $(3/2)r_s$.

/∗ ***Rules 3 and 4 are designed for the serpentine movement.*** ∗/

Rule 3: The robot checks Prefer Direction 1 for a possible movement. If the attempt to this direction is a failure, the robot executes Rule 4. Otherwise, the robot moves toward Prefer Direction 1 for a distance of $\sqrt{3}r_s$.

Rule 4: The robot checks Prefer Direction 2 for a possible movement. If the attempt to go in this direction is a failure and the deployment is not terminated, the robot will go back to the location of the previously deployed sensor and check the four rules again in order. Otherwise, the robot will move toward to Check Direction 2 for a distance of $(3/2)r_s$.

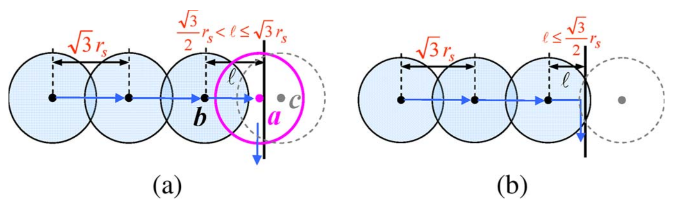
A region without any deployed sensor can be treated as a coverage hole that loses its sensing capability. Assume that the robot stays in the East state. Applying the simple serpentine movement (Rules 3 and 4), the robot will move from west to east until it encounters the right boundary and then from north to south. Thus, the most difficult situation in robot deployment is that there exists an obstacle that results to a hole in the west or north direction. However, if there is a hole in the west or north direction, the robot may apply Rules 1 and 2, which examine the west and north directions. Afterward, it moves back toward the west and north directions to deploy the sensors in the hole region. Hence, the four rules presented herein can overcome the existence of an unpredicted obstacle and achieve the objective of full-coverage deployment.

Since the obstacle-resistant serpentine deployment algorithm also considers the two check directions, the constraint that the robot should initially start its movement at the upper left corner could be released. Even though the robot initially starts its movement at the central location of the monitoring region, the robot may apply Rules 1 and 2 to move toward the west and north directions to deploy the sensors. Then, it moves in the east and south directions to achieve the objective of full-coverage deployment. The state of the robot depends on the initial location. If the robot starts its movement in the east (or west) boundary of the monitoring region, the robot will stay in the East (or West) state.

Although the aforementioned four rules may overcome the obstacles, the boundary problems still exist and may also result to a small hole at the boundary. Section IV further proposes the boundary rules that address the boundary problem.

IV. Boundary Problem and Its Handling Rules

The boundary problem is referred to the small hole at the boundary, even after the four rules proposed in Section III-B are applied. Fig. 6(a) describes the boundary problem. Assume that the robot stays in the East state and that it has recently deployed a sensor at location b. By applying Rule 3 proposed



$\sqrt{3}r_s$
$\frac{\sqrt{3}}{2}r_s < \ell \leq \sqrt{3}r_s$
ℓ
b
a
c
(a)
$\sqrt{3}r_s$
$\ell \leq \frac{\sqrt{3}}{2}r_s$
ℓ
(b)

Fig. 6. Boundary problem. The robot should make a decision on whether a sensor should be deployed at the boundary. (a) Example of a boundary problem when a robot has to deploy a sensor at the boundary. (b) Case when a robot does not have to deploy a sensor at the boundary.

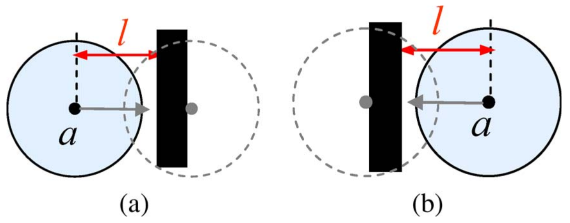


Fig. 7. Two cases in the *S-Class* boundary problem. (a) Btype 1. (b) Btype 2.

in Section III-B, the robot attempts to move for a distance of $\sqrt{3}r_s$ so that it can deploy a sensor at location c. However, the robot will encounter the right boundary before it arrives at location c, and hence, the robot moves toward the south direction without deploying a sensor at location c. In fact, whether or not the robot should deploy a sensor at the boundary depends on the boundary location. Fig. 6(b) shows a boundary situation wherein the robot does not need to deploy a sensor at the boundary. In this section, we will develop the boundary rules for deciding whether the robot should deploy a sensor at the boundary.

To make the following descriptions clear, some notations will be used and defined as follows. Let $dist(loc_1, loc_2)$ denote the distance between two locations loc_1 and loc_2. Let $s.loc$ denote the location of a deployed sensor s. Let s_1 denote the last deployed sensor before the robot encounters the boundary and s_2 denote the sensor that the robot intends to deploy after the next movement. Recall that there are six types of basic movement and a boundary or obstacle may be encountered when the robot moves using each possible type of movement. Hence, there are ten cases of boundary problem as shown in Figs. 7 and 8. Fig. 7(a) and (b) depicts two boundary cases when the obstacle is encountered if the robot applies movement types 1 and 2, respectively. Fig. 8(a) and (b) depicts the other two cases of boundary problems when the obstacle is encountered if the robot applies movement type 3. The remaining cases can be found in Fig. 8, i.e., when the obstacle is encountered if the robot applies types 4, 5, and 6. The ten cases of boundary problem can be classified into the following two classes according to movement pattern:

1) ***straight movement boundary class:*** {Btype 1, Btype 2};
2) ***corner movement boundary class:*** {Btype 3, Btype 4, Btype 5, Btype 6, Btype 7, Btype 8, Btype 9, Btype 10}.

The *straight movement boundary class* (*S-Class*) for short, consists of Btype 1 and Btype 2. In this class, the two cases have similar situations wherein the robot moves along a straight line. The other Btypes shown in Fig. 8 are included in the *corner movement boundary class* (*C-Class*) for short. The *S-Class* and *C-Class* problems will be tackled individually when developing the boundary rule.

In the following discussion, we assume that the robot has deployed sensor s_1 at location $s_1.loc$. Assume that the distance between $s_1.loc$ and the boundary is ℓ. Whether or not the robot should deploy a sensor at the boundary location mainly depends on the relationship of $s_1.loc$ and l.

A. *Boundary Rule for the S-Class Boundary Problem*

This section considers the robot moving toward the east or west direction (type 1 and type 2) but encounters a boundary or an obstacle. In the case of $l \leq (\sqrt{3}/2)r_s$, the robot does not need to deploy a sensor at the boundary. On the other hand, the robot should deploy a sensor at the boundary in the case of $l > (\sqrt{3}/2)r_s$. Fig. 6(a) depicts the scenario of $(\sqrt{3}/2)r_s < l < \sqrt{3}r_s$ wherein the robot should deploy a sensor at location a to avoid the existence of a sensing hole. Fig. 6(b) depicts the other scenario of $l \leq (\sqrt{3}/2)r_s$. The robot does not need to deploy a sensor in this case, and full coverage can still be achieved. As soon as the robot handles the obstacle or boundary problem at location a, it should return to position b and execute the obstacle-handling rules again. The following boundary rule (BRule 1) concludes the aforementioned discussion. Brule 1 will be applied when movement types 1 and 2 of the robot result to failure.

BRule 1: If $(\sqrt{3}/2)r_s < l < \sqrt{3}r_s$, the robot will deploy a sensor near the boundary. Otherwise, the robot does not need to deploy a sensor near the boundary, and it simply returns to $s_1.loc$.

B. *Boundary Rule for the C-Class Boundary Problem*

The *C-Class* boundary problem occurs when the robot applies movement types 3, 4, 5, and 6 but encounters a boundary or an obstacle. It is different from the *S-Class* boundary problem since each movement type in *C-Class* has two segments, i.e., moving to the north or south direction first and then moving to the east or west direction. The robot may encounter the obstacle either in the first or the second segment, as shown in Fig. 8. In the case of $l \leq (1/2)r_s$, the robot does not need to deploy a sensor at the boundary location while the robot moves toward the south or north direction. On the other hand, the robot should deploy a sensor at the boundary location in the case of $l > (1/2)r_s$. Since each of movement types 3, 4, 5, and 6 consists of two segments, when the robot encounters the boundary during the movement of the first segment, it checks the value l and makes the deployment decision. Then, the robot continues to move along segment 2 to complete the basic movement. Hence, the boundary problems with Btypes 3, 5, 7, and 9 can be resolved.

The remaining boundary problems in *C-Class* are Btypes 4, 6, 8, and 10, where the robot encounters the boundary at the second segment. Since the robot has completed the movement

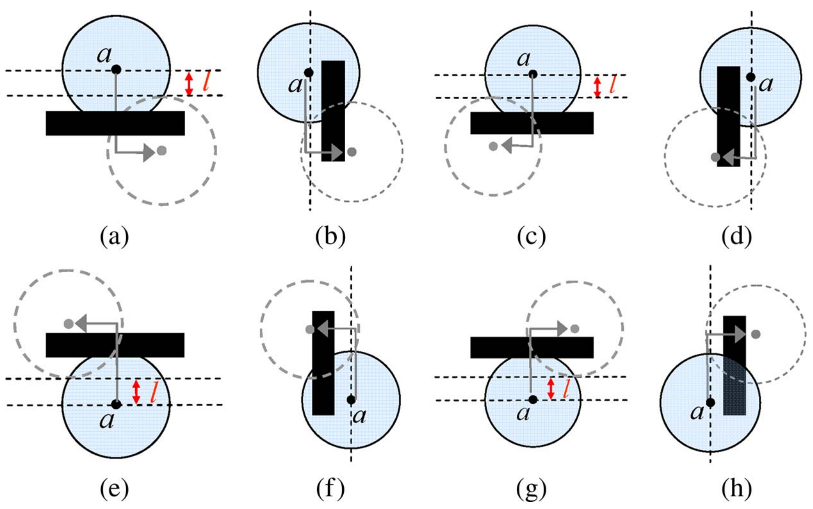


Fig. 8. Eight cases in the *C-Class* boundary problem. (a) Btype 3. (b) Btype 4. (c) Btype 5. (d) Btype 6. (e) Btype 7. (f) Btype 8. (g) Btype 9. (h) Btype 10.

of the first segment, the l value must be equal to $(3/2)r_s$. Thus, the robot has to deploy a sensor at the boundary location to achieve full coverage.

After the robot handles the boundary problem and completes the current basic movement, it should return to $s_1.loc$ and execute the obstacle-handling rules again. The following boundary rule (BRule 2) concludes the aforementioned discussion. BRule 2 will be applied when the robot applies movement type 3, 4, 5, or 6 and results to failure owing to the existence of an obstacle or a boundary.

BRule 2: If $(1/2)r_s < l < (\sqrt{3}/2)r_s$, the robot should deploy a sensor near the boundary or obstacle. Otherwise, the robot does not need to deploy any sensor, and it simply returns to $s_1.loc$.

Fig. 9(a) depicts an example of applying the proposed BRules to overcome the *C-Class* boundary problem. Assume that the robot stays in the West state and moves to location a by applying movement type 2. Once the robot deploys a sensor at location a, it applies the obstacle-resistant serpentine deployment mechanism and checks whether a hole exists in the check direction. The robot will subsequently check if a hole exists in the east, northeast, or northwest direction. If there is a hole in the northwest direction, then the robot applies movement type 5 and attempts to deploy a sensor at location b. Unfortunately, the robot encounters an obstacle on the first segment of movement type 5 while moving toward the north direction. Thus, Btype 7 of the *C-Class* boundary problem is encountered, as shown in Fig. 8(e). Since the value l satisfies the criterion $(1/2)r_s < l < (\sqrt{3}/2)r_s$, the robot will execute the BRule 2 and deploy a sensor at the location c. Afterward, the robot returns to the position a and continues to execute the obstacle-resistant serpentine deployment. The robot applies the movement type 3 and deploys a sensor at location d. A similar situation occurs at location a, and the robot detects that there is a sensing hole that exists in the northwest direction. According to the obstacle-resistant serpentine movement rules, the robot applies movement type 5 and deploys a sensor at location e. In the end, the robot achieves full-coverage deployment even though an obstacle or a boundary exists in the north direction.

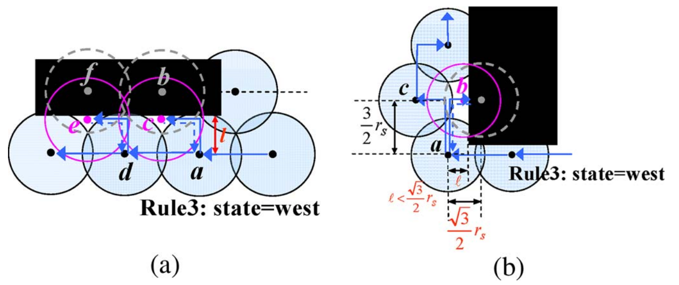


Fig. 9. Example of overcoming the *C-class* boundary problem by applying the BRules. (a) An example of the *C-Class* boundary problem. The robot applies BRule 2 to achieve the full-coverage deployment. (b) Boundary for Rule 2. (The robot moves toward Check Direction 2 by about $(3/2)r_s$ and then turns to Check Direction 1 or Prefer Direction 1 whose moving distance is not enough $(\sqrt{3}/2)r_s$.)

Fig. 9(b) depicts an example of the Btype 10 boundary problem shown in Fig. 8(h). Assume that the robot stays in the West state. The robot moves to the location a by applying movement type 2 and then deploys a sensor in that location. Then, the robot will check the check directions, including the east, northeast, and northwest directions. In this example, the robot observes that there is a hole that existed in the northeast direction. Hence, the robot applies movement type 6, with the intention of deploying a sensor at this location. However, the robot encounters the obstacle on the second segment of movement type 6, and hence, the robot applies Brule 2 to cope with the Btype 10 boundary problem shown in Fig. 8(h). The robot deploys a sensor at the location b and then returns to the location a. After that, the robot will move to location c by applying movement type 5 and deploy a sensor to achieve full coverage.

C. Putting Obstacle and Boundary Handling Rules Together

The following algorithm integrates the obstacle-resistant serpentine deployment rules and the boundary handling rules.

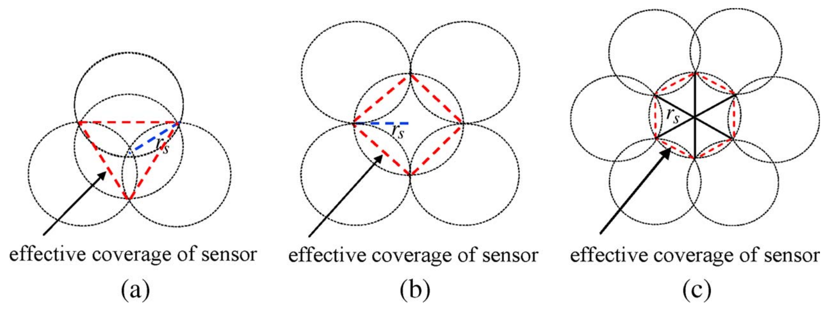

r_s
r_s
r_s
effective coverage of sensor
effective coverage of sensor
effective coverage of sensor
(a)
(b)
(c)

Fig. 10. Three ways of regular deployment and their effective coverage regions. (a) Triangle deployment. (b) Square deployment. (c) Hexagon deployment.

Algorithm: ***ORRD***

Input: A given monitoring region, which may contain obstacles. A robot that carries static sensors is located in the region.

Output: A region deployed with a near-minimal number of sensors and has achieved full coverage.

```
1.  while there is an uncovered hole do
2.    Rule1.execute
3.      if Rule1.success = false then
4.        Brule1.execute
5.        Rule2.execute
6.        if Rule2.success = false then
7.        Brule2.execute
8.        Rule3.execute
9.          if Rule3.success = false then
10.           Brule1.execute
11.           Rule4.execute
12.           if Rule4.success = false then
13.             Brule2.execute
14.           else
15.             switch the state
16.           end if
17.         end if
18.       end if
19.     end
20. end while
```

Each rule has the *execute* and *success* attributes. The execute attribute represents the robot executing this rule, while the success attribute is designed to indicate whether the movement is a success or not. As described in Section III-C2, the obstacle-handling rules have a higher priority than the serpentine movement rules. Hence, the robot initially executes Rule 1, which handles the hole region caused by the obstacle. Line 2 reflects the execution of the obstacle-handling rule. However, a boundary might be encountered during the execution of the obstacle-handling rule. This boundary could block the robot's movement and cause a movement failure. Line 4 means that the robot executes the boundary rules when the execution of Rule 1 is a failure. Lines 5, 6, and 7 are similar to lines 2, 3, and 4, respectively. In other words, the robot will initially try the obstacle-handling rule. The boundary rule will be executed in case the execution of obstacle-handling rule is a failure. Line 8 implies that the robot executes the simple serpentine movement. Similarly, as shown by lines 9 and 10, the boundary rule will be executed when the simple serpentine movement is a failure. Lines 11, 12, and 13 are similar to lines 8, 9, and 10, respectively. These three lines handle another rule for the serpentine movement. Notice that the robot should switch its state if there is success when executing Rule 4.

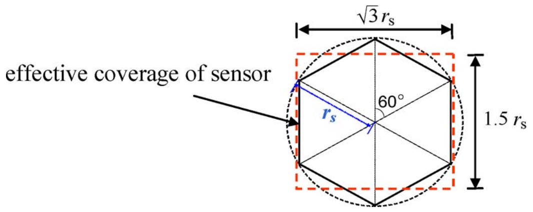


Fig. 11. Effective coverage region of hexagon deployment treated as a rectangle region that has the same area size as the hexagon region shown in Fig. 10(c).

V. Analysis and Deployment Efficiency

This section analyzes the performance efficiency of the ORRD algorithm on the optimality of the deployed network. First, a basic analysis that the given monitoring area contains no obstacle is considered to analyze the efficiency of the ORRD algorithm. Based on the basic analysis, the efficiency of the ORRD algorithm for an obstacle environment is then evaluated.

A. *Without the Obstacle Environment*

Let the *effective coverage region* of a sensor refer to the average coverage region that the sensor contributes to the monitoring area. Increasing the effective coverage of each sensor will reduce the number of sensors required to be deployed to achieve the objective of full coverage. For full coverage, there are three ways of regular deployment, namely, 1) triangle deployment, 2) square deployment, and 3) hexagon deployment, as shown in Fig. 10(a)–(c), respectively. In Fig. 10, the dotted polygon denotes the effective coverage region of each deployed sensor, where r_s denotes the sensing range. It is trivial to say that the hexagon deployment gave the best result, wherein each sensor had the largest effective coverage region. Under the hexagon deployment, the size of the effective coverage region (a hexagonal region) can be evaluated by

$$\text{Area} = \Delta \times 6 = \frac{\sqrt{3}}{4} r_s^2 \times 6 = \frac{3\sqrt{3}}{2} r_s^2.$$

To simplify the deployment efficiency analysis of ORRD, the effective coverage region of a sensor in hexagon deployment can be treated as a rectangular region with a size of $1.5r_s \times \sqrt{3}r_s$, as shown in Fig. 11, and has the same effective coverage

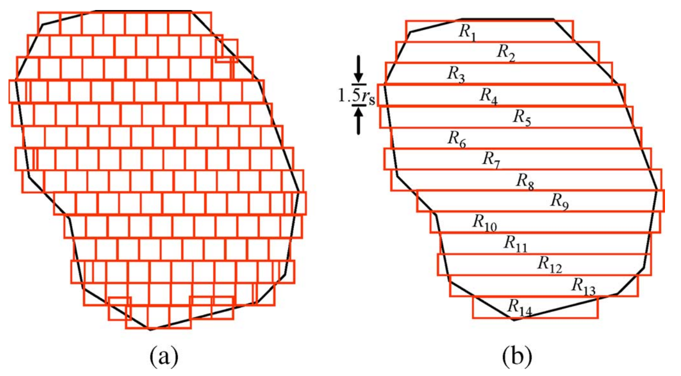


Fig. 12. Example of irregular monitoring considered in the optimality analysis for the ORRD mechanism.

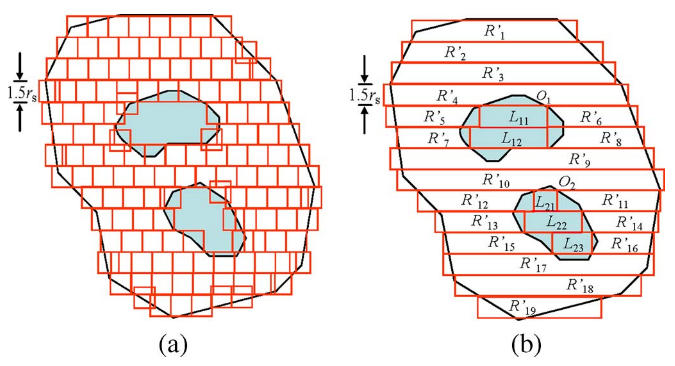


Fig. 13. Irregular monitoring region containing irregular obstacles considered for optimality analysis by the ORRD mechanism.

region size as the hexagonal region. We refer to the rectangle region shown in Fig. 11 as the optimal effective region. The optimal deployment to a given area is that which not only deploys the given monitoring area with the minimal number of sensors but also achieves full coverage. The number of sensors required by an optimal deployment is equal to the number of optimal effective regions that partition the given monitoring area.

Let L and W denote the length and width of the given monitoring area R, respectively. Fig. 12(a) shows an example of an irregular monitoring area and the deployment by applying the proposed ORRD. Since ORRD applies the horizontal movement, we assume that the monitoring region R can be partitioned into n disjoint horizontal rectangles denoted by R_i and with the width of $1.5r_s$, as shown in Fig. 12(b). Let the length of R_i be $x_i r_s$. The ideal number of sensor nodes required by optimal deployment can be derived by

$$N_{\text{ideal}}(R) = \left\lceil \sum_{i=1}^{n} \frac{\text{area of } R_i}{\text{optimal effective coverage region}} \right\rceil = \left\lceil \sum_{i=1}^{n} \frac{x_i r_s}{\sqrt{3} r_s} \right\rceil. \tag{1}$$

In the real scenario, the robot may need to deploy one more sensor as it encounters the left or right boundaries of each region R_i. Hence, the robot may deploy more sensors than with the ideal case. Equation (2) evaluates the number of sensors deployed by applying the ORRD mechanism in an average case, i.e.,

$$N_{\text{ORRD}}(R) = \sum_{i=1}^{n} \left\lceil \frac{\text{area of } R_i}{\text{optimal effective coverage region}} \right\rceil = \sum_{i=1}^{n} \left\lceil \frac{x_i r_s}{\sqrt{3} r_s} \right\rceil. \tag{2}$$

Let kr_s be the average length of all regions R_i, where k can be derived by

$$k = \frac{1}{n} \times \sum_{i=1}^{n} x_i.$$

Let the *effective ratio* of ORRD be the actual number of sensor nodes required by ORRD over the optimized number. The effective ratio of ORRD can be evaluated by the ratio of $N_{\text{ORRD}}(R)$ over $N_{\text{ideal}}(R)$, as shown in (3). Equation (3) implies that the effective ratio of ORRD decreases with the average length of regions R_i. This also indicates that applying ORRD to continually deploy a large number of sensors in horizontal movements will improve the effective ratio of ORRD. Hence, the number of sensors required by ORRD will approach the value for the ideal deployment. We have

$$\begin{aligned}
\text{Effective\_Ratio}(R) &= \frac{N_{\text{ORRD}}(R)}{N_{\text{ideal}}(R)} = \frac{\sum_{i=1}^{n} \left\lceil \frac{x_i r_s}{\sqrt{3} r_s} \right\rceil}{\left\lceil \sum_{i=1}^{n} \frac{x_i r_s}{\sqrt{3} r_s} \right\rceil} \\
&\leq \frac{\sum_{i=1}^{n} \left\lceil \frac{x_i r_s}{\sqrt{3} r_s} \right\rceil}{\sum_{i=1}^{n} \frac{x_i r_s}{\sqrt{3} r_s}} \leq \frac{\sum_{i=1}^{n} \left(\frac{x_i}{\sqrt{3}} + 1 \right)}{\sum_{i=1}^{n} \frac{x_i r_s}{\sqrt{3} r_s}} \\
&= \frac{\frac{(k+\sqrt{3})n}{\sqrt{3}}}{\frac{kn}{\sqrt{3}}} = 1 + \frac{\sqrt{3}}{k}.
\end{aligned} \tag{3}$$

B. *With the Obstacle Environment*

This subsection analyzes the efficiency of ORRD in a given monitoring area that contains obstacles. Fig. 13(a) shows an example of an irregular monitoring area with multiple obstacles and the deployment of ORRD. Initially, we do not consider the existence of any obstacle, and we assume that the monitoring area R is partitioned into n disjoint horizontal rectangles, denoted by R_i with a width of $1.5r_s$, as shown in Fig. 12(b). Let the length of R_i be $x_i r_s$. Equations (1) and (2) can be applied to evaluate the numbers of sensor nodes required by the ideal deployment and the ORRD mechanism, respectively.

Assume that there are m obstacles $O_1, O_2, \ldots, O_m$ existing in the irregular monitoring area. Each obstacle O_i can cover w_i disjoint horizontal rectangles denoted by L_{ij} and with a width of $1.5r_s$, where $1 \leq j \leq w_i$, as shown in Fig. 13(b). Let the length of L_{ij} be $l_{ij}r_s$. Let h_ir_s be the average length of the horizontal rectangle L_{ij}, where h_i can be derived by

$$h_i = \frac{1}{w_i} \times \sum_{j=1}^{w_i} l_{ij}.$$

The total length of horizontal rectangles covered by all obstacles can be estimated by

$$l_{\text{obstacles}} = \sum_{i=1}^{m} h_i w_i. \tag{4}$$

Since the obstacles may further partition the horizontal rectangles, we assume that the number of disjoint horizontal rectangles outside the obstacles is t, which is a number larger than n. Let the t horizontal rectangles be denoted by R'_i, where $1 \le i \le t$, as shown in Fig. 13(b), and the length of R'_i be x'_ir_s. Let kr_s be the average length of all regions R'_i. The value k can be estimated by the difference between the total lengths of x_i and $l_{\text{obstacles}}$ divided by the number of disjoint horizontal rectangles t, as derived by

$$k = \frac{1}{t} \times \left(\sum_{i=1}^{n} x_i - \sum_{i=1}^{m} h_i w_i \right) \tag{5}$$

$$\text{Effective\_Ratio}(R) = \frac{N_{\text{ORRD}}(R)}{N_{\text{ideal}}(R)} \le \frac{\sum_{i=1}^{m} \left\lceil \frac{x'_i r_s}{\sqrt{3} r_s} \right\rceil}{\left\lceil \sum_{i=1}^{m} \frac{x'_i r_s}{\sqrt{3} r_s} \right\rceil}$$

$$\le 1 + \frac{t}{\sum_{i=1}^{n} x_i - \sum_{i=1}^{m} h_i w_i}. \tag{6}$$

According to (3), the effective ratio of ORRD can be determined by k, which is the value of the average length of all regions R_i over r_s. Hence, in a monitoring region with obstacles, the effective ratio of ORRD can be evaluated by substituting (5) into (3), as shown in (6). In this case, the obstacles will reduce the number of continually deployed sensors in the horizontal movements and, hence, reduce the average length of all regions R'_i. The small number of continually deployed sensors in the horizontal movement will increase the difference in the numbers of deployed sensors between ORRD and the ideal deployment, i.e., increasing the value of the Effective_Ratio(R) of ORRD. Equation (6) implies that increasing the number of obstacles or the total area of obstacles will reduce the chance of continually deployed sensors in the horizontal movements and, hence, decrease the value of the Effective_Ratio(R) of ORRD.

VI. Performance Study

This section examines the performance study of the developed obstacle-resistant deployment mechanism. The proposed ORRD mechanism is compared with previous work in [15] and is referred to as CED.

TABLE V
SIMULATION PARAMETERS

Parameter	**Value**
Communication range	40m
Sensing range	20m
Packet transmission cost	0.075J/s
Packet reception cost	0.030J/s
Idle cost	0.025J/s
Maximum energy consumption in motes	324J/hr
Total initial energy	32400J (100hr)

A. System Model

Table V lists the parameters values that refer to the typical parameters in the Berkeley motes [19]. The robot is assumed to be equipped with a compass and a constant number of Berkeley motes. The total energy and the speed of the robot are 64 800 J and 3 m/s, respectively. The mobility cost is set to 8.267 J/m. The experimental environment is described below. The network size is $400 * 400$ m^2. The starting location of the robot is at the top left spot. The simulation result is obtained from an average of 100 independent runs.

The proposed ORRD is compared with the CED [15] and LRV [18] in terms of the number of deployed sensors, the coverage percentage, and the energy consumption in a WSN environment with or without obstacles. The four movement policies, namely, 1) random order, 2) cross order, 3) line order, and 4) circle order, as proposed by LRV [18], are implemented. Two types of obstacles, namely, 1) regular obstacles and 2) irregular obstacles, are considered in the experiment. Regular obstacles are randomly generated from among five different shapes, as shown in Fig. 14. The process for generating an irregular obstacle is more complicated than for regular ones. In the experiment, an irregular obstacle is composed of 6, 9, or 12 unit squares, with each unit square having an edge length of r_s, where r_s denotes the sensing range. The generation of an irregular obstacle with size k consists of $k+1$ phases, where k is 6, 9, or 12 unit squares. Initially, an irregular obstacle contains only one unit square. Therefore, the obstacle boundary is composed of four edges. In its second phase, an edge is randomly selected from the four edges, and an additional unit square is attached to the selected edge. As a result, the boundary of the irregular obstacle then becomes composed of six edges. Let the boundary of an irregular obstacle in the ith phase be composed of n_i edges. In the $(i+1)$th phase, an edge is randomly selected from the n_i edges, and an additional unit square is attached to the selected edge. This process of obstacle generation is executed phase by phase until k phases are reached. As a result, the boundary of the generated irregular obstacle with size k becomes composed of n_k edges. Fig. 15(a) shows an example of an irregular obstacle composed of nine unit squares. Finally, in the $(k+1)$th phase, each of the n_k edges is replaced by one arc randomly selected from the arcs shown in Fig. 15(b). The following gives the details of the edge replacement process. The arcs shown in Fig. 15(b) are classified into two sets: 1) horizontal and 2) vertical. In the case where

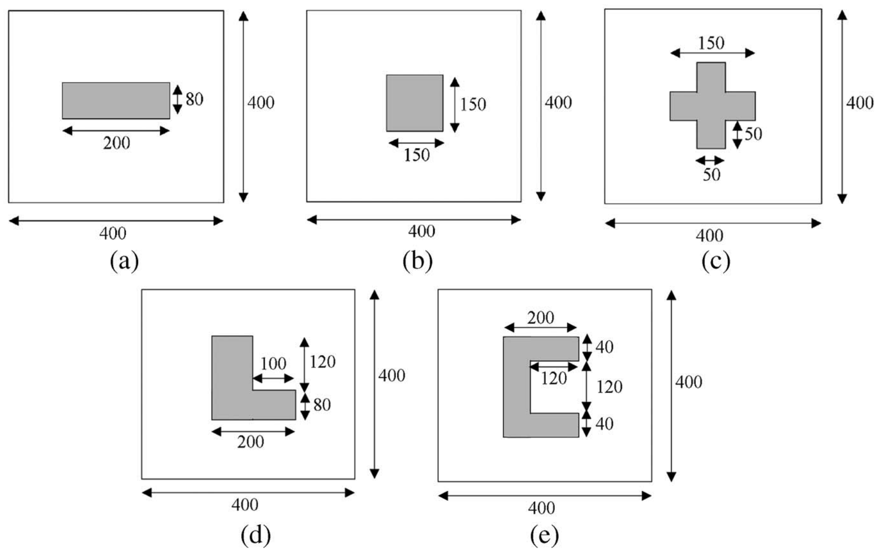


Fig. 14. Shapes of regular obstacles considered in the simulation. (a) Rectangle. (b) Square. (c) X-shape. (d) L-shape. (e) C-shape.

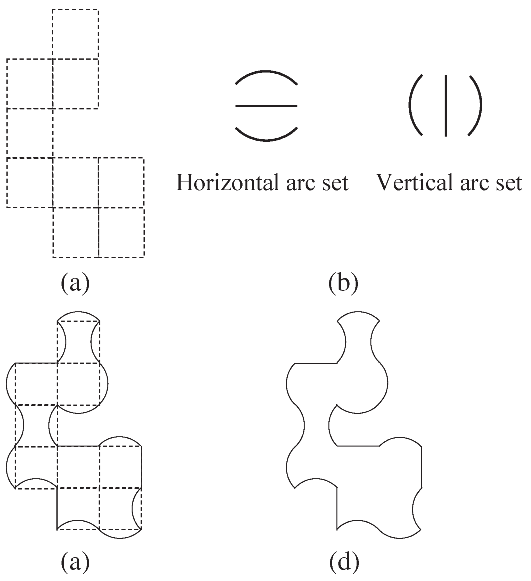


Fig. 15. Generation of an irregular obstacle. (a) Example of an irregular obstacle with nine unit squares. (b) Six possible arcs that replace the boundaries of an irregular obstacle. (c) Boundaries of an irregular obstacle replaced by the arcs shown in (b). (d) Final version of the generated irregular obstacle with nine unit squares.

the edge of the constructed obstacle is a horizontal edge, it is replaced by the arc randomly selected from the horizontal arc set; otherwise, it is replaced by the arc selected from the vertical arc set. Fig. 15(c) depicts the generated irregular obstacle with nine unit squares after the obstacle boundary is replaced by arcs, while Fig. 15(d) shows the final version of the generated irregular obstacle with nine unit squares.

B. Number of Deployed Sensors

The performance of ORRD is examined in terms of the number of deployed sensors in the environment with/without obstacles. Fig. 16(a) depicts the screenshot of the development executed by ORRD in an environment without an obstacle. In Fig. 16(a), the number labeled in each sensor denotes the deployment order. The robot deploys a total of 174 sensor nodes to achieve the full sensing coverage. In most cases, the ORRD mechanism deploys the minimum number of sensor nodes, but the whole monitoring area is fully covered. Fig. 16(b) depicts the deployment by applying ORRD in an environment containing a C-shape obstacle. It is observed that the robot can efficiently overcome the obstacle and deploys 163 sensors, which is also the minimal number of sensors required to achieve the objective of full coverage.

Table VI presents the number of deployed sensors when applying the ORRD, CED, and LRV mechanisms. An obstacle with a regular-corner shape is considered in the simulation environment. According to the predefined priority of the south, west, north, and east directions, the CED deploys a sensor when the robot moves out of the sensing region of the deployed sensor. To reduce the impact of the obstacles, LRV determines the movement direction according to the four policies of random, cross, line, and circle orders. By comparison, LRV outperforms the CED in most cases in terms of the number of deployed sensors. Since the CED and LRV deploy a sensor by negotiating with the closest deployed sensors, the newly deployed sensor may have significant redundant sensing regions with the sensing regions of the neighboring sensors. The proposed ORRD significantly reduces the redundant coverage regions and, hence, outperforms the CED and LRV in all cases in terms of the number of deployed sensors.

Let *coverage percentage* denote the ratio of the given monitoring region covered by the deployed sensors. Fig. 17 compares ORRD, CED, and LRV in terms of the coverage percentage in the environment that contains an obstacle with different regular-corner shapes. LRV-random has poor performance in all cases because the random-walk movement strategy is applied. By comparison, the coverage percentage of the CED is higher than that of LRV in all cases, except in the environment

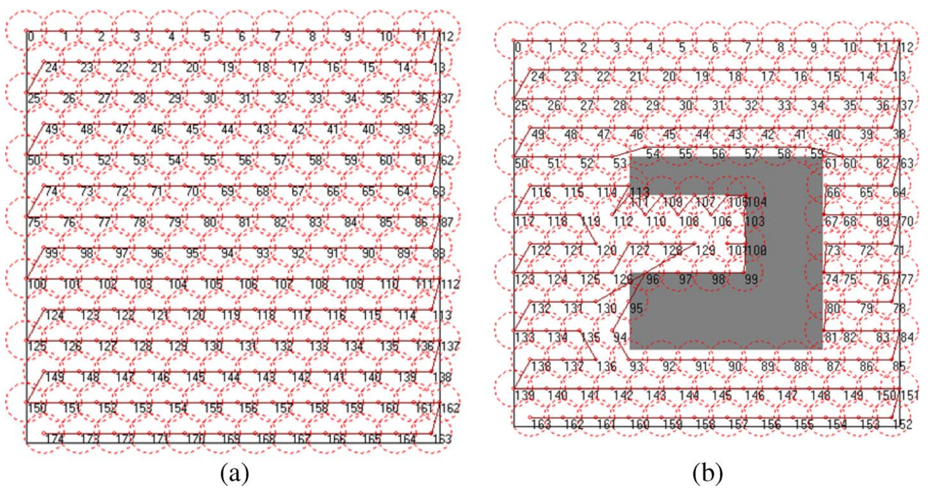


Fig. 16. Screenshots of the deployment by applying the proposed ORRD mechanism. (a) Screenshot of the deployment by applying ORRD in an environment without an obstacle. (b) Screenshot of the deployment by applying ORRD in an environment containing a C-shape obstacle.

TABLE VI
COMPARISONS BETWEEN THE ORRD, CED, AND LRV SCHEMES IN TERMS OF THE NUMBER OF DEPLOYED SENSOR NODES IN AN ENVIRONMENT CONTAINING A REGULAR OBSTACLE WITH VARIOUS SHAPES

	No obstacle	Rectangle obstacle	Square obstacle	X-shape obstacle	L-shape obstacle	C-shape obstacle
ORRD	174	166	162	171	160	163
CED	230	211	224	228	204	215
LRV-Random	242	198	231	216	234	221
LRV-Cross	210	200	197	199	220	204
LRV-Line	208	194	185	189	208	219
LRV-Circle	214	187	181	182	202	190

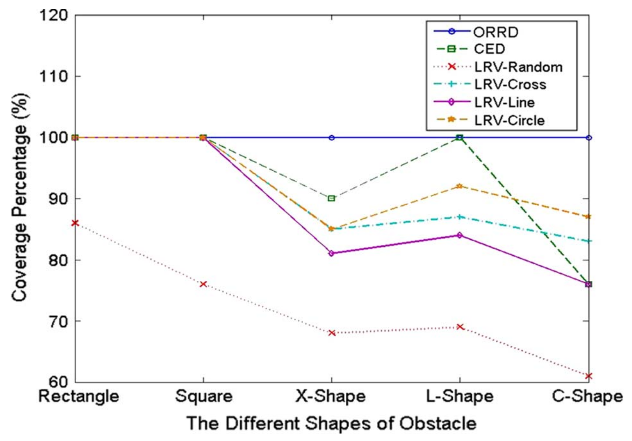


Fig. 17. Comparison of coverage percentage in the environment that contains a regular obstacle with various shapes.

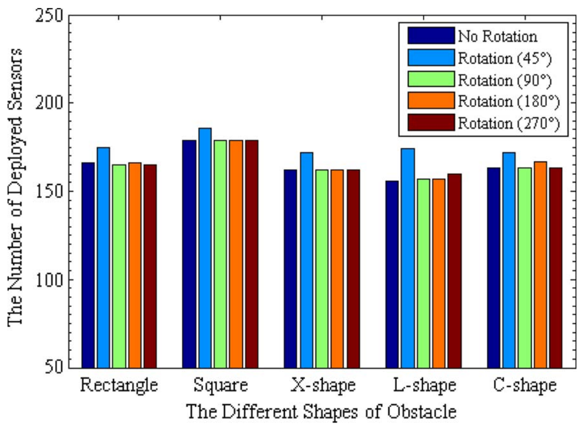


Fig. 18. ORRD deploying a constant number of deployed sensors even though the five-shape obstacles are rotated with 45°, 90°, 180°, and 270° in the environment.

that contains the C-shape obstacle. In general, ORRD keeps the coverage percentage above 99% and outperforms CED and LRV in all cases.

Fig. 18 rotates the regular obstacles with five shapes to measure the number of deployed sensors by applying ORRD. The obstacle is rotated by 45°, 90°, 180°, and 270°, and then, the ORRD mechanism is applied to investigate the impacts of the different obstacle shapes on the ORRD performance. As shown in Fig. 18, ORRD deployed more sensors in the environment where the obstacle is rotated by 45°. This is because ORRD applies the boundary rules to avoid the existence of sensing hole and, thus, deploys more sensors near the boundaries of the obstacle. In general, the proposed ORRD deploys a similar number of sensors and can overcome various shapes of obstacles.

Fig. 19 compares the proposed ORRD with the ideal case in terms of the number of deployed sensors when the WSN contains irregular obstacles. The number of irregular obstacles ranges from one to three, and each obstacle is composed of six, nine, or 12 unit squares. Since the shapes and sizes of the

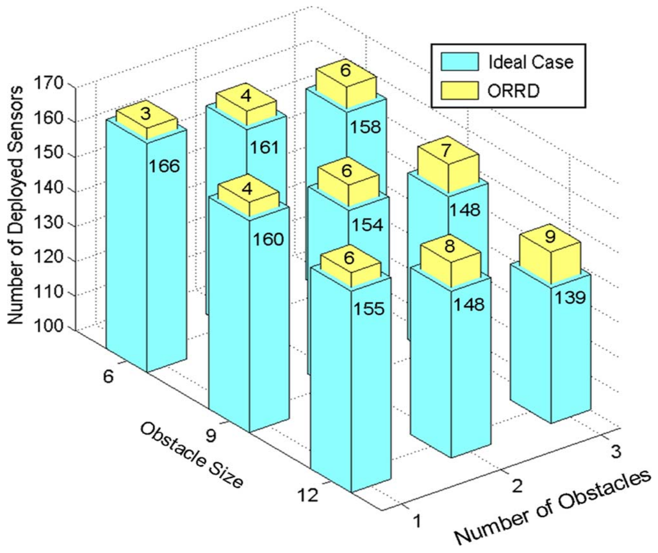


Fig. 19. Impact of irregular obstacles on the number of deployment sensors.

irregular obstacles are known in the ideal case, the minimal number of required sensors to fully cover the monitoring region can be obtained. However, ORRD does not have the obstacle information; hence, it will apply the boundary rules to additionally deploy sensors near the obstacle boundaries to achieve full coverage. As a result, ORRD deploys more sensors than the ideal case for irregular obstacles. In Fig. 19, the numbers labeled in the ideal case represent the number of sensors required in the ideal case, while the numbers labeled in ORRD denote the number of sensors that need to be additionally deployed by applying the proposed ORRD, as compared with the ideal case. In general, both of them have a similar result, namely, the number of deployed sensors decreases along with the size or the number of obstacles. The reason for this is because there is no need to deploy any sensor within the obstacle region, which is enlarged along with the size and the number of obstacles. The difference between ORRD and the ideal case in terms of the number of deployed sensors highly depends on the total boundary length of irregular obstacles. The total boundary length increases when the number of obstacles grows or when the sizes of the obstacles are increased. Compared with the ideal case, the number of sensors additionally deployed by ORRD therefore increases, along with the size and the number of obstacles.

C. Deployment Efficiency

Deployment efficiency, which is used to measure the efficiency of the deployment task, is the ratio of the ideal and the actual number of deployed sensors, i.e.,

$$\text{deployment efficiency} = \frac{\text{ideal number of sensor deployment}}{\text{real number of sensor deployment}}. \quad (7)$$

A deployment efficiency of a deployment mechanism that is close to 1 indicates that the deployment mechanism approaches the optimal deployment. Fig. 20 compares the ORRD, CED,

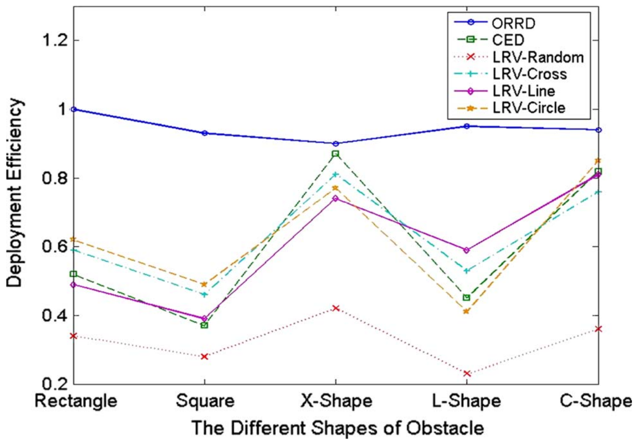


Fig. 20. Comparison of ORRD, CED and LRV in terms of deployment efficiency.

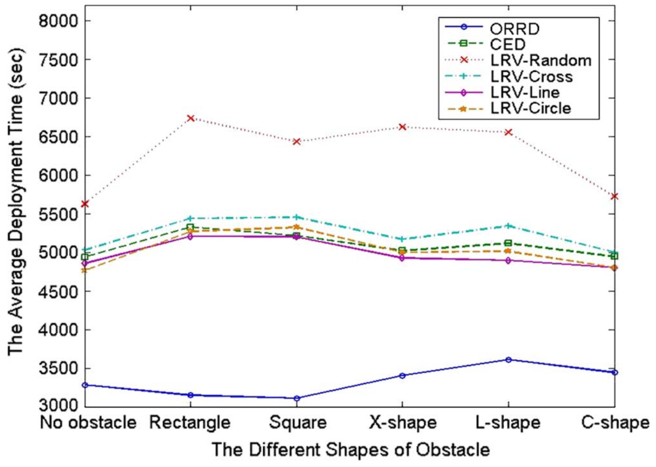


Fig. 21. Average deployment time after applying the ORRD, CED, and LRV in the environment containing multiple regular obstacles with different shapes.

and LRV mechanisms in terms of the deployment efficiency and the number of sensors. The monitoring area contains obstacles with specific shapes, including rectangle, square, X-shape, L-shape, and C-shape. The deployment efficiencies of the CED and LRV, which significantly change with the obstacle shapes, have average values of 0.6 and 0.58, respectively. In general, ORRD keeps an almost constant deployment efficiency value greater than 0.95 and outperforms the other deployment mechanisms in all cases. By applying the proposed ORRD, the number of deployed sensors approaches the ideal number.

D. Deployment Time

In the simulation, the deployment time refers to the time required for the robot to achieve full coverage or a predefined percentage of coverage. A shorter deployment time indicates that the robot deploys sensors by traveling an efficient path and, hence, allows the WSN to start working earlier. Each simulation result is obtained from the average of deployment times required for the environment containing regular- and round-corner obstacles. Fig. 21 compares ORRD, CED, and LRV in terms of the deployment time to achieve the full coverage.

TABLE VII
ENERGY CONSUMPTION OF ORRD, CED, AND LRV

	No obstacle	Rectangle obstacle	Square obstacle	X-shape obstacle	L-shape obstacle	C-shape obstacle
ORRD	48.438KJ	48.010KJ	47.435KJ	51.716KJ	53.311KJ	52.320KJ
CED	72.915KJ	71.825KJ	69.845KJ	74.343KJ	76.142KJ	74.422KJ
LRV-Random	81.529KJ	77.395KJ	75.149KJ	85.984KJ	89.526KJ	84.866KJ
LRV-Cross	74.256KJ	73.258KJ	70.968KJ	76.168.KJ	77.968KJ	76.449KJ
LRV-Line	71.029KJ	68.539KJ	65.387KJ	72.695KJ	74.558KJ	73.298KJ
LRV-Circle	69.985KJ	67.522KJ	71.556KJ	73.115KJ	76.665KJ	73.007KJ

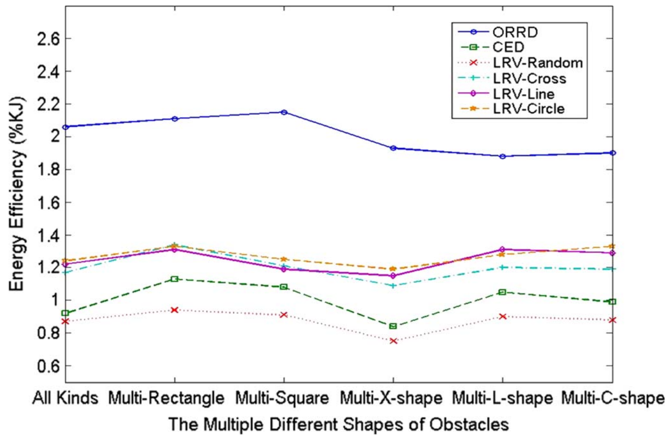


Fig. 22. Comparison of ORRD, CED and LRV in terms of energy efficiency within a specific time period.

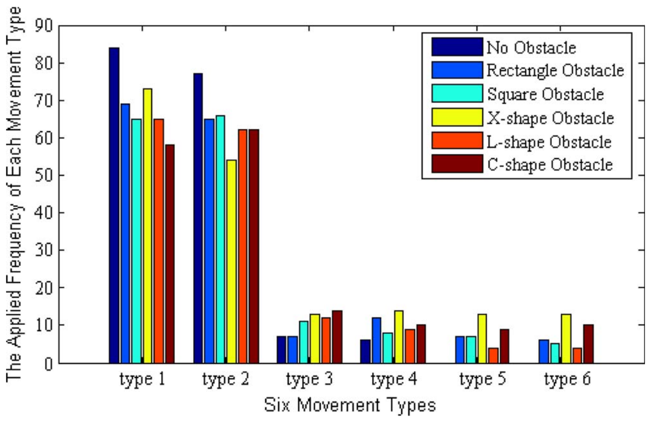


Fig. 23. Frequencies of six movement types applied by the robot that executes ORRD in different environments.

LRV-random has the longest deployment time in all cases because no efficient movement strategy is applied. The CED, LRV-line, and LRV-circle apply predefined movement rules and spend similar deployment times in the environment without any obstacle. However, they spend more time than ORRD to achieve the full coverage. In the environment that contains obstacles, the CED and LRV results in several holes that were subsequently healed after a relatively long period of time. The proposed ORRD efficiently overcomes different shapes of obstacles and achieves full coverage. Consequently, the proposed ORRD outperforms the CED and LRV in terms of deployment time.

E. Energy Efficiency

The total energy consumption of the robot is used to evaluate the efficiency of the robot's moving trajectory. The energy consumption is increased with the length of the robot's movement. Table VII summaries the total energy consumption of the robot by applying ORRD, CED, and LRV in the environment containing obstacles with different shapes. The LRV-random mechanism deploys sensors without a movement strategy and, hence, consumes more energy than the other mechanisms in all cases. The CED and the other three mechanisms proposed in LRV have similar performance. ORRD outperforms all the other compared mechanisms in terms of energy consumption. The major reason is that ORRD deploys sensors and overcomes the obstacles with an efficient trajectory.

Energy efficiency, which is used to measure the effectiveness of the robot's movement, is the ratio of the final coverage percentage and the total energy consumption within a predefined time period, i.e.,

$$\text{energy efficiency} = \frac{\text{final coverage percentage}}{\text{total energy consumption}}. \quad (8)$$

Fig. 22 compares the energy efficiency of ORRD, CED, and LRV. The multi-X-shape obstacles raised the boundary problems, and ORRD applies BRules 1 and 2 to achieve the full coverage. However, the application of BRules 1 and 2 also introduces the redundant movements and reduces the energy efficiency. As a result, ORRD has the lowest energy efficiency in the environment containing the multi-X-shape obstacles.

On the other hand, ORRD has the best efficient movement when the obstacles are with the square or rectangular shape. By comparison, ORRD achieves better energy efficiency than the CED and LRV in all cases. The major reason is that the CED and LRV have many redundant movements when the environment contains obstacles. Another factor that lowers the performance of the CED and LRV is that they cannot overcome the obstacles, and thus, the deployment results in many sensing holes within the predefined time period.

F. Applied Frequency of Movement Types and Rules

ORRD classifies the basic movement into six patterns. Fig. 23 depicts the usages of the six movement patterns. In the experiments, movement types 1 and 2 are used most frequently. In the environment without any obstacle, the robot always moves toward the east or west until the boundary is encountered. Hence, movement types 1 and 2 are frequently

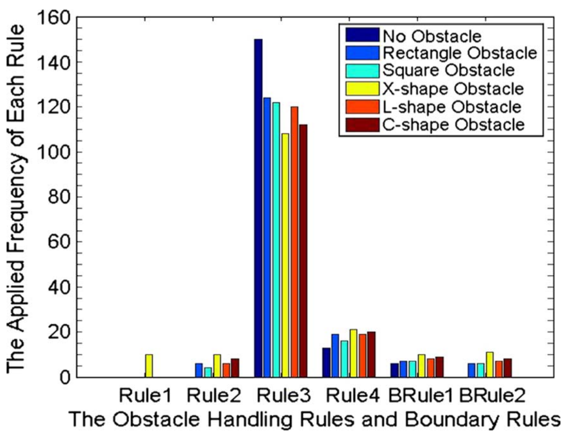


Fig. 24. Applied frequency of each rule proposed in ORRD in an environment containing a regular obstacle with various shapes.

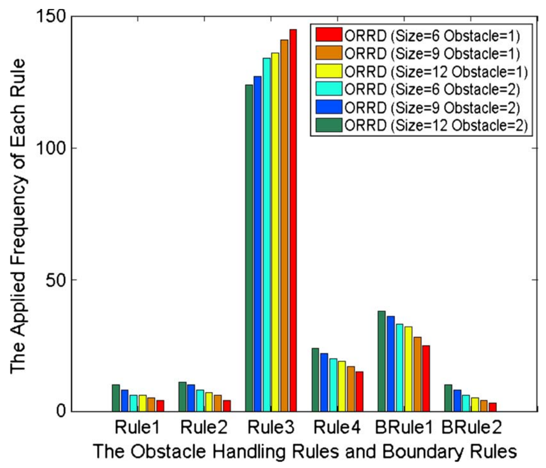


Fig. 25. Applied frequency of each rule proposed in ORRD in environments containing one or two irregular obstacles of sizes six, nine, or 12 unit squares.

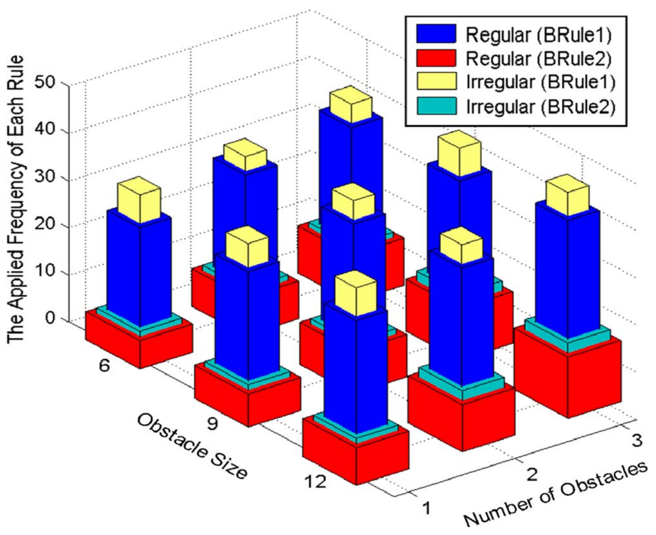


Fig. 26. Applied frequencies of BRules 1 and 2 for the environment containing irregular obstacles.

applied. In the environment with obstacles, the robot still attempts to move in a serpentine manner, i.e., moving toward the east or west direction. To avoid the existence of sensing hole, movement types 3, 4, 5, and 6 are used to check whether a hole exists in the direction opposed to the current state. Thus, such movement types will actually be applied only when the obstacle is encountered. In particular, movement types 3, 4, 5, and 6 are frequently used when the shapes of the obstacles are irregular. For example, these types of movements are frequently used by the robot when the obstacles are X-shape, L-shape, and C-shape.

Fig. 24 measures the usage of the proposed four rules in the obstacle-resistant mechanism and the two boundary rules. Different shapes of obstacles are considered in the environment. Recall that Rules 1 and 2 are designed for overcoming obstacles, while Rules 3 and 4 are designed for the serpentine movement. In the environment without an obstacle, ORRD applies Rules 3 and 4 to achieve full coverage. As soon as different shapes of obstacles exist in the environment, Rules 1 and 2 are applied. In addition, a big hole might be surrounded by several boundaries, and several small holes might exist near the boundary. Hence, the robot applies the proposed four boundary rules to cope with the boundary problem. The usage of the BRule highly depends on the location and the shape of the obstacle. Recall that the BRule is applied when the following condition holds:

$$\begin{cases} \frac{\sqrt{3}}{2}r_s < l < \sqrt{3}r_s, & \text{for an S-Class boundary problem} \\ \frac{1}{2}r_s < l < \frac{\sqrt{3}}{2}r_s, & \text{for a C-Class boundary problem.} \end{cases}$$

As shown in Fig. 22, the applied frequency of BRule 1 that handles the failure in movement types 1 and 2 is larger than that of BRule 2 in case that the shapes of obstacle are rectangular, square, L-shaped, and C-shaped.

Fig. 25 measures the usage of the proposed rules in ORRD in an environment that contains multiple irregular obstacles with various sizes. The number of irregular obstacles is set at one or two, while the size of each obstacle is varied by six, nine, or 12 unit squares. Since Rule 3 is mainly designed for the serpentine movement, which can only be applied in regions without obstacles, the applied frequency of Rule 3 decreases along with the size and the number of the obstacles. Rule 4 is applied when switching between the "West" and "East" states whenever boundaries of obstacles or monitoring regions are encountered by the robot. Similarly, the BRule 1 is applied only when the boundaries of obstacles and monitoring regions are encountered by the robot. As a result, the applied frequencies of Rule 4 and BRule 1 increase along with the number and size of the obstacles. Compared with BRule 1, BRule 2, Rule 1, and Rule 2 are applied less frequently because they are designed for handling special situations that appear less frequent.

Fig. 26 investigates the impact of irregular obstacles on the applied frequencies of BRule 1 and BRule 2. The number of obstacles ranges from 1 to 3, and the sizes of each obstacle are set to six, nine, and 12 unit squares. In general, the applied frequency of BRule 1 is more than that of BRule 2, regardless of whether the obstacle shape is regular or irregular. The proposed BRule 1 is applied for overcoming irregular obstacles more frequently than regular obstacles because the robot might additionally deploy more sensors at locations near the boundaries of irregular obstacles.

VII. Conclusion

This paper has proposed the ORRD mechanism, which overcomes obstacles, deploys a near-optimal number of sensors over the monitoring region, and likely achieves full coverage. We initially proposed a deployment policy and the serpentine movement for deploying a minimal number of sensors in the environment without an obstacle. Then, we proposed four obstacle-handling rules to cope with the sensing holes problem due to the existing obstacles. In addition, we presented the boundary problem and proposed two boundary rules to cope with it. Eventually, the proposed ORRD integrates the deployment policy, the serpentine movement policy, the obstacle-resistant rules, and the boundary-handling rules. Performance results reveal that the proposed ORRD significantly outperforms the existing CED and LRV mechanisms in terms of the number of deployed sensors, the coverage percentage, the required deployment time, and the energy consumption of the robot.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Comput. Netw.*, vol. 38, no. 4, pp. 393–422, Mar. 2002.

[3] M. S. Pan, C. H. Tsai, and Y. C. Tseng, "Emergency guiding and monitoring applications in indoor 3D environments by wireless sensor networks," *Int. J. Sensor Netw.*, vol. 1, no. 1/2, pp. 2–10, Sep. 2006.

[4] W. Liang and Y. Liu, "Online data gathering for maximizing network lifetime in sensor networks," *IEEE Trans. Mobile Comput.*, vol. 6, no. 1, pp. 2–11, Jan. 2007.

[5] A. Boukerche, *Handbook of Algorithms for Wireless Networking and Mobile Computing*. London, U.K.: Chapman & Hall, 2005.

[6] A. Boukerche and X. Fei, "A coverage-preserving scheme for wireless sensor network with irregular sensing range," *Ad Hoc Netw.*, vol. 5, no. 8, pp. 1303–1316, Nov. 2007.

[7] A. Boukerche, X. Fei, and R. B. Araujo, "An optimal coverage-preserving scheme for wireless sensor networks based on local information exchange," *Comput. Commun.*, vol. 30, no. 14/15, pp. 2708–2720, Oct. 2007.

[8] B. Carbunar, A. Grama, J. Vitek, and O. Carbunar, "Redundancy and coverage detection in sensor networks," *ACM Trans. Sensor Netw.*, vol. 2, no. 1, pp. 94–128, Feb. 2006.

[9] H. Gupta, Z. H. Zhou, S. R. Das, and Q. Gu, "Connected sensor cover: Self-organization of sensor networks for efficient query execution," *IEEE/ACM Trans. Netw.*, vol. 14, no. 1, pp. 55–67, Feb. 2006.

[10] N. Heo and P. K. Varshney, "Energy-efficient deployment of intelligent mobile sensor networks," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 35, no. 1, pp. 78–92, Jan. 2005.

[11] S. Chellappan, X. Bai, B. Ma, D. Xuan, and C. Xu, "Mobility limited flip-based sensor networks deployment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 2, pp. 199–211, Feb. 2007.

[12] G. Wang, G. Cao, and T. L. Porta, "Movement-assisted sensor deployment," *IEEE Trans. Mobile Comput.*, vol. 5, no. 6, pp. 640–652, Jun. 2006.

[13] G. Wang, G. Cao, T. L. Porta, and W. Zhang, "Sensor relocation in mobile sensor networks," in *Proc. 24th Annu. INFOCOM*, Miami, FL, Mar. 2005, pp. 2302–2312.

[14] M. A. Batalin and G. S. Sukhatme, "Efficient exploration without localization," in *Proc. ICRA*, Taipei, Taiwan, May 2003, pp. 2714–2719.

[15] M. A. Batalin and G. S. Sukhatme, "Coverage, exploration and deployment by a mobile robot and communication network," *Telecommun. Syst.—Special Issue Wireless Sensor Networks*, vol. 26, no. 2–4, pp. 181–196, Jun. 2004.

[16] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization in distributed sensor networks," *ACM Trans. Embed. Comput. Syst.*, vol. 3, no. 1, pp. 61–91, Feb. 2004.

[17] Y. C. Wang, C. C. Hu, and Y. C. Tseng, "Efficient deployment algorithms for ensuring coverage and connectivity of wireless sensor networks," in *Proc. IEEE WICON*, 2005, pp. 114–121.

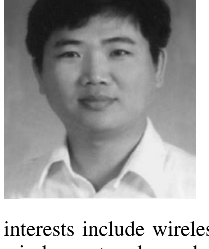
[18] M. A. Batalin and G. S. Sukhatme, "The design and analysis of an efficient local algorithm for coverage and exploration based on sensor network deployment," *IEEE Trans. Robot.*, vol. 23, no. 4, pp. 661–675, Aug. 2007.

[19] J. Hill and D. Culler, "A wireless embedded sensor architecture for system-level optimization," Comput. Sci. Dept., Univ. Calif., Berkeley, Berkeley, CA, 2002. Tech. Rep.

Chih-Yung Chang (M'01–A'01) received the Ph.D. degree in computer science and information engineering from National Central University, Taoyuan, Taiwan, in 1995.

In 1997, he joined the faculty of the Department of Computer and Information Science, Aletheia University, Tamsui, Taiwan, as an Assistant Professor, where he was the Chair of the Department of Computer and Information Science from August 2000 to July 2002. In August 2002, he joined the Department of Computer Science and Information Engineering (CSIE), Tamkang University, Tamsui, as an Associate Professor and where he is currently a Full Professor. He served as an Associate Guest Editor for the *Journal of Information Science and Engineering* in 2008, the *Journal of Internet Technology* in 2004 and 2008, and the *Journal of Mobile Multimedia* in 2005 and as a Member of the Editorial Board of *Tamsui Oxford Journal of Mathematical Sciences* from 2001 to 2008 and the *Journal of Information Technology and Applications* in 2008. His current research interests include wireless sensor networks, Bluetooth radio networks, ad hoc wireless networks, and WiMAX broadband technologies.

Dr. Chang is a member of both the IEEE Computer and Communications Societies.



Chao-Tsun Chang was born in Taipei, Taiwan. He received the Ph.D. degree in computer science and information engineering from the National Central University, Taoyuan, Taiwan, in 2006.

In 2006, he joined the faculty of the Department of Information Management, Hsiuping Institute of Technology, Taichung, Taiwan, as an Assistant Professor. Over the past ten years, he has directed 11 research projects, including three National Science Council (NSC) projects and eight information system development projects. His current research interests include wireless sensor networks, Bluetooth radio networks, ad hoc wireless networks, and mobile computing.

Dr. Chang is a member of both the IEEE Computer and IEEE Communications Societies.

Yu-Chieh Chen received the B.S. degree from Ming Chuan University, Taipei, Taiwan, in 2005 and the M.S. degree from Tamkang University, Tamsui, Taiwan, in 2007, both in computer science and information engineering. Since 2007, he has been working toward the Ph.D. degree with the Department of Computer Science and Information Engineering, Tamkang University.

He has received numerous scholarships in Taiwan and participated in many wireless sensor networking projects. His research interests include wireless sensor networks, ad hoc wireless networks, mobile/wireless computing, and WiMAX.

Hsu-Ruey Chang received the B.S. and Ph.D. degrees in computer science and information engineering from Tamkang University, Tamsui, Taiwan, in 2003 and 2007, respectively.

He is currently a Software Engineer with Inventec, Taipei, Taiwan. He has received numerous scholarships in Taiwan and participated in many Bluetooth and wireless sensor networking projects. He has published extensively in the wireless networking area. His research interests are wireless sensor networks, ad hoc wireless networks, and mobile/wireless computing, including both theoretic results and algorithm design.