An Obstacle-Free and Power-Efficient Deployment Algorithm for Wireless Sensor Networks

Chih-Yung Chang, Jang-Ping Sheu, *Senior Member, IEEE*, Yu-Chieh Chen, and Sheng-Wen Chang

***Abstract*—This paper proposes a robot-deployment algorithm that overcomes unpredicted obstacles and employs full-coverage deployment with a minimal number of sensor nodes. Without the location information, node placement and spiral movement policies are proposed for the robot to deploy sensors efficiently to achieve power conservation and full coverage, while an obstacle surrounding movement policy is proposed to reduce the impacts of an obstacle upon deployment. Simulation results reveal that the proposed robot-deployment algorithm outperforms most existing robot-deployment mechanisms in power conservation and obstacle resistance and therefore achieves a better deployment performance.**

***Index Terms*—Deployment, obstacle, placement, robot, wireless sensor networks (WSNs).**

I. Introduction

RECENT advances in digital electronics, microprocessors, and wireless communication have made sensors smaller, low powered, and cheaper to manufacture [1]. Due to these attractive characteristics, wireless sensor networks (WSNs) are now widely applied to many applications, which include environmental monitoring, tracking, precision agriculture, military surveillance, smart homes, and so on [2]–[7]. WSNs are large-scale distributed systems composed of a large number of sensors and a few sinks. Sensors are responsible for collecting the sensed information and sending them to sinks in a multihop manner. A sink is the interface between the sensors and users, executing tasks that include accepting the user's command, delivering the data request criteria to the sensors, collecting the interested data from the sensors, and integrating them to provide the user with advanced uses.

Developing a good deployment algorithm is one of the most important issues for an efficient WSN. In the literature, existing deployment algorithms can be classified into three categories: stationary sensor, mobile sensor, and mobile robot. Several random deployment schemes [8], [9] have been proposed for the deployment of stationary sensors. The random deployment scheme is simple and easy to implement. However, to ensure full coverage, its number of deployed sensors is extremely larger than the actual required number of sensors. The random deployment of stationary sensors may result in an inefficient WSN wherein some areas have a high density of sensors while others have a low density. Areas with high density increase hardware costs, computation time, and communication overheads, whereas areas with low density may raise the problems of coverage holes or network partitions. Other works [10]–[13] have discussed deployment using mobile sensors. Mobile sensors first cooperatively compute for their target locations according to their information on holes after an initial phase of random deployment of stationary sensors and then move to target locations. However, hardware costs cannot be lessened for areas that have a high density of stationary sensors deployed.

Another deployment alternative [14]–[17] is to use the robot to deploy static sensors. The robot explores the environment and deploys a stationary sensor to the target location from time to time. The robot deployment can achieve full coverage with fewer sensors, increase the sensing effectiveness of stationary sensors, and guarantee full coverage and connectivity. Aside from this, the robot may perform other missions such as hole-detection, redeployment, and monitoring. However, unpredicted obstacles are a challenge of robot deployment and have a great impact on deployment efficiency. One of the most important issues in developing a robot-deployment mechanism is to use fewer sensors for achieving both full coverage and energy-efficient purposes even if the monitoring region contains unpredicted obstacles.

Obstacles such as walls, buildings, blockhouses, and pillboxes might exist in the outdoor environment. These obstacles significantly impact the performance of robot deployment. A robot-deployment algorithm without considering obstacles might result in coverage holes or might spend a long time executing a deployment task. In the literature, Batalin and Sukhatme [14] assume that the robot is equipped with a compass which makes it aware of its movement direction. A robot movement strategy that uses deployed sensors to guide a robot's movement as well as sensor deployment in a given area is proposed. Although the proposed robot-deployment scheme likely achieves the purpose of full coverage and network connectivity, it does not, however, take into account the obstacles. The next movement of the robot is guided from only the nearest sensor node, raising problems of coverage holes or overlapping in the sensing range as the robot encounters obstacles. Aside from this, during robot deployment, all deployed sensors stay in an

Manuscript received February 13, 2007; revised May 24, 2008. First published April 17, 2009; current version published June 19, 2009. This work was supported in part by the National Science Council, Taiwan. This paper was recommended by Associate Editor A. Ollero.

C.-Y. Chang, Y.-C. Chen, and S.-W. 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; swchang@wireless.cs.tku.edu.tw).

J.-P. Sheu is with the Department of Computer Science and Information Engineering, National Tsing Hua University, Hsinchu 30013, Taiwan (e-mail: sheujp@cs.nthu.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/TSMCA.2009.2014389

active state in order to participate in guiding tasks, resulting in an inefficiency in power consumption.

To handle obstacle problems, a previous research [17] has proposed a centralized algorithm that uses global obstacle information to calculate for 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, which makes the developed robot-deployment mechanism useful only in limited applications.

This work aims to develop an obstacle-free robot-deployment algorithm. Unpredicted obstacles are taken into consideration in the proposed mechanism so full coverage can be likely achieved with the deployment of a minimal number of sensors. Moreover, most deployed sensors may stay in a sleep state to reduce power consumption during robot deployment. Simulation results show that the proposed algorithm significantly reduces the total number of deployed stationary sensors, achieves full coverage, and saves on energy consumption.

The rest of this paper is organized as follows. Section II first reviews related work; then, the network model and the basic concept of this paper are given. Section III gives the details of the robot-deployment algorithm. The node placement and spiral movement policies are proposed without considering the existence of obstacles. Section IV presents the obstacle handling mechanism and discusses the energy conservation issue. The performance study is presented in Section V. Finally, conclusions are drawn in Section VI.

II. Related Work and Basic Concepts

This section initially describes related work on robot deployment. Later on, the network environment and basic concepts of this paper are introduced with examples.

A. Related Work

Compared with random deployment, using the robot to stepwise deploy static sensors in a specific region can give full sensing coverage with fewer sensors. Previous research [14] assumes that the robot is equipped with a compass and is able to detect obstacles. Each sensor has a communication range r_c and a sensing range r_s. To guide the robot's movement, each deployed sensor maintains a time duration for each direction, whether south, east, north, or west, that the robot did not visit. The longer the time length, the higher priority the direction is. When the robot intends to make a movement decision, it communicates with the closest deployed sensor, queries the time length of each direction, and then selects one direction with the highest priority to be the direction of its next movement.

Although the robot-deployment algorithm developed in [14] likely achieves the purpose of full coverage and network connectivity, the next movement of the robot is guided by only one sensor, resulting in it taking a long time to achieve full coverage and requiring more sensors due to a big overlapping area. As shown in Fig. 1(a), assume that sensors s_k and s_j were previously deployed in the monitoring area and sensor s_i is the most recently deployed sensor. According to the guiding

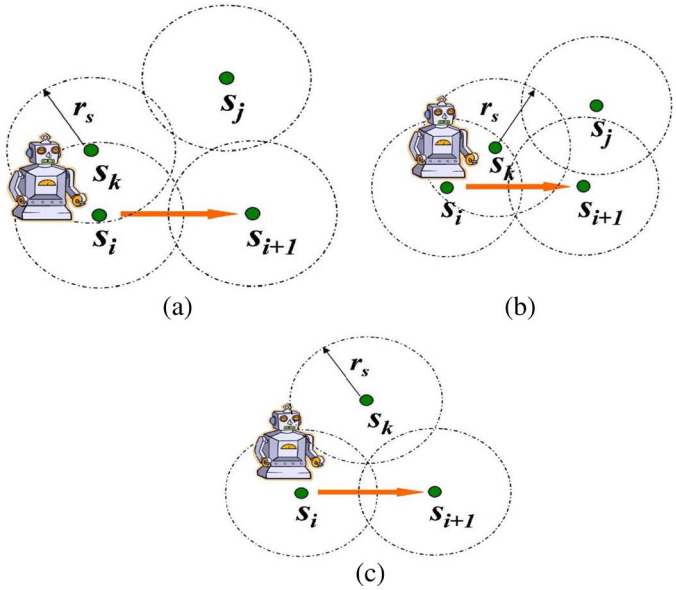


Fig. 1. Drawback of the algorithm in [14] is due to the situation where the next movement of the robot is guided by only one sensor. (a) Coverage hole. (b) Many overlapping areas. (c) Improved deployment.

of sensor s_i, the robot will move east and deploy sensor s_{i+1}, resulting in a hole this time even though sensor s_k is in the communication range of the robot. A sensor will be deployed to the hole after a long period. Fig. 1(b) shows another example. Assume that the distance between s_i and s_k is smaller than r_s. The robot only guided by s_i may deploy a sensor s_{i+1} to the location that overlaps a lot of the sensing regions of s_k and s_j, requiring more sensors to be deployed on the whole target region. This paper develops an efficient robot-deployment algorithm. The robot will be guided by s_i and s_k at the same time to achieve full coverage and network connectivity purposes using fewer deployed sensors. The improved deployment in this case is shown in Fig. 1(c). The overlapping area among the sensing ranges of s_i, s_{i+1}, and s_k is minimal, achieving full coverage by using fewer sensors.

In addition, the algorithm proposed in [14] does not consider power conservation. Since all sensors have no knowledge on when the robot will revisit them, all sensors should be active and therefore consume significant energy. Power conservation should be taken into consideration in developing the robot-deployment algorithm wherein most deployed sensors may stay in sleep mode during the robot-deployment process. Aside from this, the robot movement policy proposed in [14] also leaves holes whenever an obstacle is encountered. Fig. 2 shows the robot starting to deploy sensor s_1 and then moving south to deploy sensors s_2 and s_3, leaving a hole in the target region.

This paper aims to develop a robot-deployment algorithm that has the following characteristics. The robot deploys sensors in a way that fewer sensors are deployed but are likely to achieve full coverage. As for power conservation, most deployed sensors can stay in sleep mode to conserve energy. In addition, the developed deployment algorithm can resist obstacles so that fewer sensors need to be deployed to achieve full sensing coverage even if there are obstacles in the monitoring area.

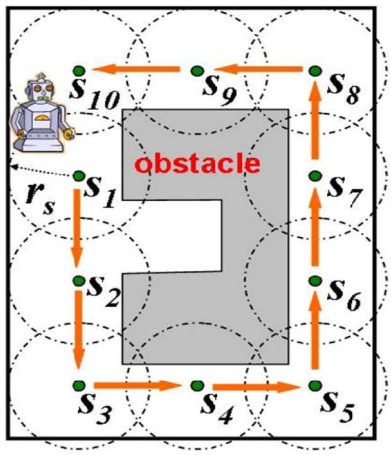


Fig. 2. Robot movement policy leaves a hole when an obstacle is encountered.

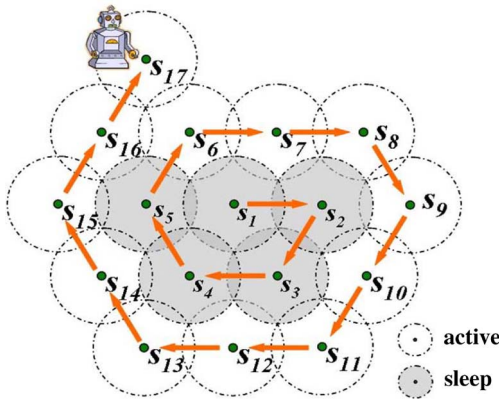


Fig. 3. Sensor placement and spiral movement policies.

B. Network Environment and Protocol Overview

1) Network Environment: This paper assumes that the robot is aware of its own location information. This assumption can be achieved if the robot is equipped with both GPS and compass modules which provide the current location and moving direction of the robot, respectively. Although the GPS localization system might have an inaccuracy range, the other localization system based on the compass module could help estimate the current location of the robot according to the movement direction provided by the compass as well as the distance, which can be calculated based on the rotation rate of the robot's wheels. We also assume that the monitoring field is a 2-D plane where there may exist obstacles in the field and that the robot is embedded with a radio system that enables it to communicate with deployed sensors in a wireless manner. Aside from this, the robot is also able to discover unknown obstacles as it moves closer to them. Let symbol s_m denote a sensor, $0 \leq m \leq N$, where N is the number of sensors required for full coverage. The communication range is denoted by r_c, while the sensing range is denoted by r_s, and they both satisfy the relation $r_c \geq \sqrt{3}r_s$. Given two locations p and q in the monitoring field, symbol $d(p, q)$ represents the distance of p and q. In the following, an example is given to illustrate the basic idea of the proposed robot-deployment protocol.

2) Node Placement and Spiral Movement Policies: To deploy fewer sensors to achieve full coverage, a node placement policy is used, as illustrated in the following. If the distance between any two sensors is equal to $\sqrt{3}r_s$, then there should be no hole and only minimal overlap in the area. The node placement policy is that the robot deploys a sensor for every distance of $\sqrt{3}r_s$. In addition to the placement policy, a spiral movement policy is adopted as a strategy for the robot movement. That is, the robot would move clockwise in a spiral way. As shown in Fig. 3, sensors s_i, $1 \leq i \leq 17$, have been deployed by the robot that uses the spiral movement policy. According to the node placement and spiral movement policies, a robot can deploy a sensor every $\sqrt{3}r_s$ distance in a spiral manner. Thus, the deployed region achieves full coverage without holes, and the overlapping sensing areas among the deployed sensors are minimal.

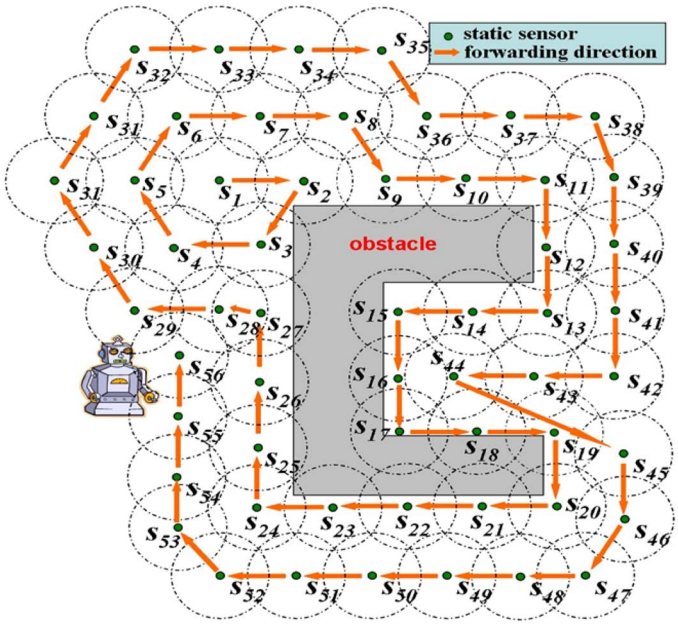


Fig. 4. Basic deployment concept when the robot encounters an obstacle.

3) Obstacle Resistance: The developed robot-deployment protocol likely achieves a full-coverage deployment even if the global information of obstacles is unavailable. When the robot encounters obstacles, it moves around them to deploy sensors and eliminate the impact of obstacles on deployment. During the execution of the deployment task, the robot switches between steady and obstacle states, depending on whether it comes across obstacles. The state is maintained by the robot and the deployed sensors. Initially, the robot stays in a steady state. As shown in Fig. 4, the robot uses the spiral movement policy to deploy sensors s_i, $1 \leq i \leq 9$. As the robot encounters obstacles, it marks s_9 as an initiator and switches to obstacle state. After that, the robot surrounds the obstacle, deploys sensors s_i, $10 \leq i \leq 27$, and sets them to be in the obstacle state. When the robot encounters sensor s_3, which is the first encountered sensor in a steady state, the robot switches from obstacle to steady state and deploys sensors s_i, $28 \leq i \leq 36$, according to the node placement and spiral movement policies. Once the robot encounters the obstacle or any sensor staying in the obstacle state, it treats those sensing regions of sensors s_i, $10 \leq i \leq 27$, that stay in

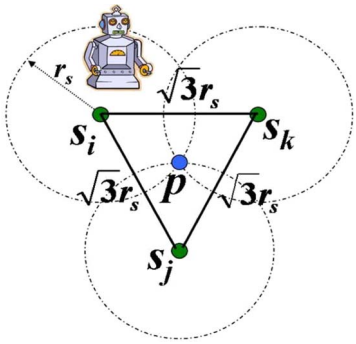


Fig. 5. Optimal deployment of sensors s_i, s_j, and s_k.

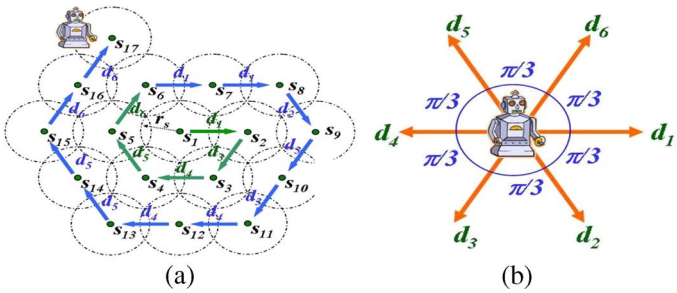


Fig. 6. Movement trajectory of the robot. (a) Spiral movement. (b) Six cardinal directions.

obstacle state as virtual obstacles and uses a similar way to overcome these obstacles. In the obstacle state, the robot maintains the locations of the deployed sensors in Stack. Whenever the robot moves to a location that is surrounded by the deployed sensors, it pops the location from Stack and moves to it until the robot is not surrounded by the deployed sensors. Fig. 4 shows an example to illustrate how the robot copes with the dead-end problem. Let $pos(s)$ denote the location of a deployed sensor s. When the robot deploys sensor s_{44}, it subsequently pops locations $pos(s_{43})$ and $pos(s_{42})$ from Stack and deploys a new sensor s_{45}. Finally, the robot deploys sensors around the obstacle until the deployed sensors cover the entire monitoring area.

III. Robot-Deployment Algorithm

This section describes the deployment algorithm that achieves full coverage with fewer sensors. The algorithm mainly consists of a *node placement policy* and a *spiral movement policy*.

A. Node Placement Policy

Let s_i, s_j, and s_k denote three deployed sensors as shown in Fig. 5. The sensing region of each sensor is denoted by a dotted circle. Let r_s and r_c denote the sensing range and communication range of each sensor, respectively. To deploy fewer sensors while maintaining full sensing coverage, the overlapping of the sensing region of sensors s_i, s_j, and s_k should be minimal. Therefore, the three sensors should be deployed in a way as shown in Fig. 5, wherein sensors s_i, s_j, and s_k intersect exactly at a point p. With this deployment, the distance between two neighboring sensors is $\sqrt{3}r_s$. To prevent the neighboring sensors from losing communication, this paper assumes that the communication range r_c is equal or larger than $\sqrt{3}r_s$. The following gives a node placement policy.

1) Node Placement Policy: The robot deploys a sensor for every $\sqrt{3}r_s$ distance.

When the robot deploys a sensor in the monitoring region, it provides the deployed sensor with the current location information. Upon receiving the location information, the deployed sensor maintains this information for later use.

B. Spiral Movement Policy

To overcome the obstacle, the robot adopts the spiral movement policy. As shown in Fig. 6(a), the robot moves clockwise in a spiral way and deploys sensors s_i, $1 \leq i \leq 7$, every $\sqrt{3}r_s$. To present the *spiral movement policy*, several terms are given in the following.

Definition: Cardinal Direction: In a spiral movement, the robot moves in one of six cardinal directions, denoted by d_1, d_2, d_3, d_4, d_5, and d_6, as shown in Fig. 6(b). Every two adjacent directions have an included angle of $\pi/3$.

Definition: Level and Round: In a spiral movement, the trajectory of a robot movement can be viewed as a layered hexagon where level 0 consists of node s_1 and level 1 consists of nodes s_2 up to s_7, and so on, as shown in Fig. 6(a). Nodes that lie on direction line d_5 are referred to as the *starting nodes* of each layer. The *round* is defined by the movement of a robot starting from the starting node to the next starting node.

As shown in Fig. 6(a), nodes s_1, s_6, and s_{17} are the starting nodes of rounds 0, 1, and 2, respectively. The first round of the spiral movement consists of the trajectory moving from node s_1 to node s_6. The second round of the spiral movement consists of the trajectory moving from node s_6 to node s_{17}.

Definition: Direction Sequence D_k: A direction sequence D_k of a spiral movement consists of movement directions in the kth round.

As shown in Fig. 6(a), the direction sequence $\boldsymbol{D}_1$ is $\{d_1, d_3, d_4, d_5, d_6\}$. The direction sequence of $\boldsymbol{D}_2$ is $\{d_1, d_1, d_2, d_3, d_3, d_4, d_4, d_5, d_5, d_6, d_6\}$.

In the kth round, the robot moves according to the subsequent directions in D_k. For example, in the first round of spiral movement, the robot moves according to the direction sequence $D_1 = \{d_1, d_3, d_4, d_5, d_6\}$. That is, the robot moves in direction d_1 for a distance of $\sqrt{3}r_s$ and, after, deploys a sensor. The robot then moves in direction d_3 for a distance of $\sqrt{3}r_s$ and then deploys another sensor. Following the movement directions listed in D_1, as the robot moves in the last direction d_6 and deploys a sensor, the movement of the first round is finished. The robot will then start the movement of the second round according to the direction sequence $D_2 = \{d_1, d_1, d_2, d_3, d_3, d_4, d_4, d_5, d_5, d_6, d_6\}$. Following the direction sequence in each round, a spiral movement will be achieved. In the following, the rules for generating a direction sequence for a spiral movement are proposed. Let $\boldsymbol{d}_i^{\boldsymbol{k}}$ denote k consecutive appearances of $\boldsymbol{d}_i$ in D_k. The direction sequence D_k of the kth round in the spiral movement

can be automatically generated by the robot according to the following:

$$D_k = \left\{d_1^k, d_2^{k-1}, d_3^k, d_4^k, d_5^k, d_6^k\right\}, \qquad k \in Z^+.$$

For example, the robot can generate the movement direction sequence D_k using

$$\begin{aligned} k=1, \quad & D_1 = \{d_1, d_3, d_4, d_5, d_6\} \\ k=2, \quad & D_2 = \{d_1, d_1, d_2, d_3, d_3, d_4, d_4, d_5, d_5, d_6, d_6\} \\ k=3, \quad & D_3 = \{d_1, d_1, d_1, d_2, d_2, d_3, d_3, d_3, d_4, d_4, d_4, \\ & \qquad d_5, d_5, d_5, d_6, d_6, d_6\} \text{ and so on.} \end{aligned}$$

Therefore, the robot will move according to the following spiral movement policy.

Spiral Movement Policy: The robot moves a distance of $\sqrt{3}r_s$ in a specific direction according to the direction sequence D_k in the kth round and then deploys a sensor, where

$$D_k = \left\{d_1^k, d_2^{k-1}, d_3^k, d_4^k, d_5^k, d_6^k\right\}, \qquad k \in Z^+.$$

IV. Obstacle Handling

This section introduces the obstacle handling mechanism that overcomes unknown obstacles in order to achieve full-coverage deployment. Two types of states are used to distinguish whether the robot encounters obstacles, namely, the steady and obstacle states. In the steady state, the robot moves and deploys sensors following the aforementioned spiral movement and sensor deployment policies. When the robot encounters an obstacle, it switches from steady to obstacle state and moves and deploys sensors according to an obstacle surrounding movement policy which is introduced in the following to reduce the negative impacts of an obstacle upon deployment.

A. Obstacle Surrounding Movement Policy

To reduce the impact of an obstacle upon deployment, the *obstacle surrounding movement policy* is employed in this section. When the robot encounters obstacles, it switches from steady to obstacle state. In the obstacle state, the robot adopts the obstacle surrounding movement policy, in which the robot moves clockwise around the obstacle and deploys sensors to have full coverage. Some terms are introduced first to make the illustration clear. The *guiding sensor* is the most recently deployed sensor. Note that the guiding sensor is closest to the robot. The *reference sensor* refers to the deployed sensor that is second closest to the robot. It is observed that the distances between any pair of three locations, including the expected location, the location of the guiding sensor, and the location of the reference sensor, are exactly $\sqrt{3}r_s$ apart. Let the *initiator* denote the last guiding sensor that guides the robot to encounter the obstacle. Let the *leaving sensor* denote those sensors that are nearby the obstacles and are deployed in steady state. As shown in Fig. 7, sensor s_9 is the initiator, and sensor s_3 is the leaving sensor.

To implement the obstacle surrounding movement policy, the robot moves along the obstacle in a clockwise direction.

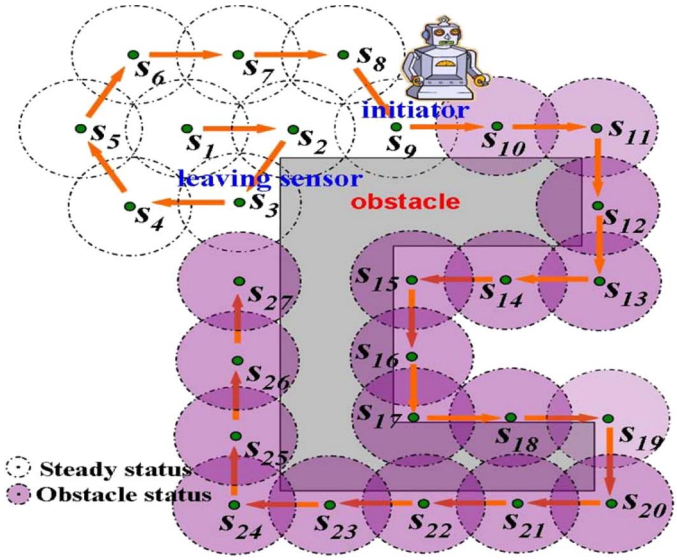


Fig. 7. Obstacle surrounding movement policy.

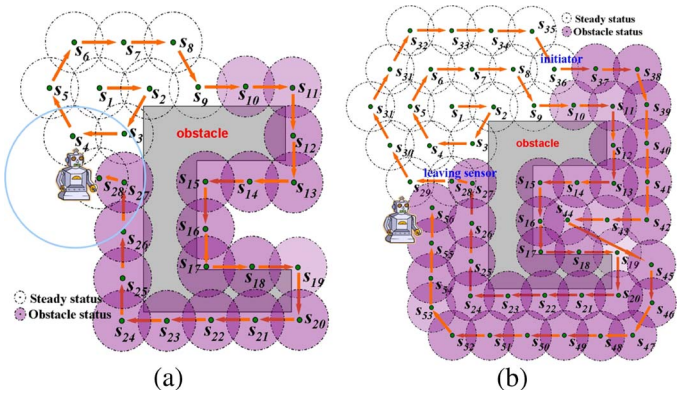


Fig. 8. Example wherein the robot applies the obstacle surrounding movement policy to overcome obstacles. (a) Robot applies the *obstacle surrounding movement policy* in obstacle state. (b) Robot treats the sensors with obstacle state as virtual obstacles and switches from steady and obstacle state to overcome obstacles.

More specifically, when the robot encounters the obstacle, it always moves around the obstacle in a clockwise direction until a leaving sensor is found. Take the example shown in Fig. 7. When the robot encounters the obstacle, it switches to the obstacle state and selects the guiding sensor s_9 as its initiator. Applying the obstacle surrounding movement policy, the robot then deploys sensors s_{10}–s_{27} around the obstacle. As the robot deploys s_{27}, it communicates with the leaving sensor s_3. This indicates that the robot has deployed sensors around the obstacle. Sensors that have been deployed by the robot in obstacle state are considered virtual obstacles, so that later, the robot may also adopt the obstacle surrounding movement policy to deploy sensors around these virtual obstacles (Fig. 8).

The following gives the details on how the robot in the obstacle state calculates the expected location for deploying its next sensor. The ultrasonic or laser sensor equipped on the robot is able to detect the distance between the farthest point of an obstacle boundary and the robot. For example, as shown in Fig. 9, the robot can detect that the distance to the farthest irregular boundary is k centimeters. Therefore, the

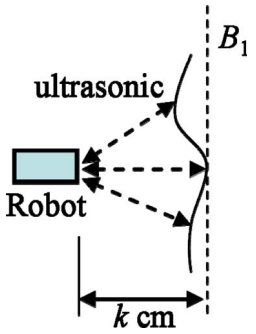


Fig. 9. Robot uses an ultrasonic sensor or a laser sensor to detect the distance of the farthest point of the obstacle boundary and the robot. Robot treats an irregular boundary as a regular boundary B_1.

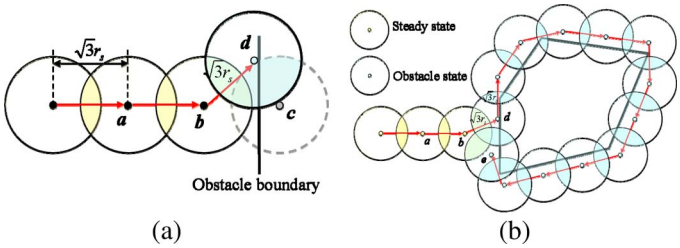


Fig. 10. Example that illustrates the details of the obstacle surrounding movement policy. (a) Example of the boundary problem wherein the robot needs to deploy a sensor at the boundary of an obstacle. (b) Case wherein the robot deploys sensors to surround the obstacle.

robot can treat an irregular boundary as a regular boundary B_1 and determine the expected location for deploying a new sensor.

Fig. 10(a) shows an example to illustrate how the robot deploys a sensor when it encounters an obstacle boundary. Assume that the robot has recently deployed a sensor s_b at location b and intends to move a distance of $\sqrt{3}r_s$ so that it can deploy a sensor at location c. However, the robot encounters an obstacle boundary before it arrives at location c. The robot then switches from steady to obstacle state, notifying sensor s_b to play the role of a leaving sensor and trying to move along the obstacle boundary in a clockwise direction until it arrives at a location, for example, d, where the distance between locations b and d is $\sqrt{3}r_s$. The robot then deploys another sensor s_d at location d. Afterward, the robot continues to move along the obstacle boundary in a clockwise direction and deploys a sensor every $\sqrt{3}r_s$ distance. Fig. 10(b) shows the result of deploying a number of sensors around an obstacle. Whenever the robot deploys a sensor s_e at location e and is aware that it is within the communication range of the leaving sensor s_b, it selects s_b and s_a to play the roles of guiding and reference sensors, respectively, and then adjusts its location for deploying its next sensor. The adjustment procedure will be presented in a later section. After this, the robot switches from obstacle to steady state and deploys its next sensor at the adjusted location.

B. Switching From Obstacle to Steady State

When the robot encounters a leaving sensor, it switches to steady state and tries to apply the spiral movement policy. The robot broadcasts a message to find its guiding and reference sensors. Upon receiving the message, the sensors in steady state then detect the signal strength of the robot and send information on the signal strength back to the robot. After, the robot selects two sensors with the highest signal strengths to be its guiding and reference sensors. The robot adjusts its location and deploys a sensor to the expected location.

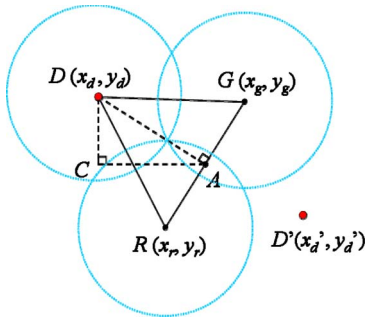


Fig. 11. Robot determines a location for deploying its next sensor from candidate locations D and D' as it switches from obstacle to steady state.

The following gives the procedure for calculating the adjusted location. When the robot encounters a leaving sensor, it switches its state from obstacle to steady and selects guiding and reference sensors. Recall that each deployed sensor maintains its location information. The robot then adjusts its location for deploying its next sensor based on the locations of the guiding and reference sensors. As shown in Fig. 11, let $G(x_g, y_g)$ and $R(x_r, y_r)$ be the locations of the guiding and reference sensors, respectively, and let the location for deploying the next sensor be $D(x_d, y_d)$. Next, we discuss how to calculate $D(x_d, y_d)$ based on the information of $G(x_g, y_g)$ and $R(x_r, y_r)$. In Fig. 11, the circle represents the sensing range of each sensor. The optimal deployment location for the next sensor is $D(x_d, y_d)$ where

$$|DG| = |DR| = |GR| = \sqrt{3}r_s. \tag{1}$$

The equation of line $\overline{GR}$ can simply be calculated by

$$\overline{GR} : \frac{y_r - y_g}{x_r - x_g}x - y + y_r - \frac{y_r - y_g}{x_r - x_g}x_r.$$

Next, we intend to calculate the coordinates of location C so that the value of x_d can be derived accordingly. The coordinates of location C can be derived based on the slope and length of line $\overline{AD}$. Since line $\overline{AD}$ is perpendicular to line $\overline{GR}$, the slope and length of line $\overline{AD}$ are derived by

$$slope_{\mathrm{AD}} = \frac{x_g - x_r}{y_r - y_g} \qquad |AD| = \left(\frac{3}{2}\right) r_s. \tag{2}$$

From the fact that

$$slope_{\mathrm{AD}} = \frac{x_g - x_r}{y_r - y_g} = \frac{|CD|}{|AC|} \qquad |CD|^2 + |AC|^2 = |AD|^2$$

we have the following:

$$[(x_g - x_r)a]^2 + [(y_r - y_g)a]^2 = \left(\frac{3}{2}r_s\right)^2. \tag{3}$$

According to (3), the value of a is

$$a = \frac{3r_s}{2\sqrt{(x_g - x_r)^2 + (y_r - y_g)^2}}. \quad (4)$$

The value of x_d can therefore be derived based on (3) and (4):

$$x_d = \frac{x_r + x_y}{2} - a(x_g - x_r).$$

Similarly, the value of y_d can be derived using

$$y_d = \frac{y_r + y_g}{2} + a(y_r - y_g).$$

Note that the two locations D and D' both satisfy (1), as shown in Fig. 11. The robot selects a location D_{next} closest to the last deployed sensor from candidates D and D'.

Fig. 8 shows an example of the robot switching from obstacle to steady state. In Fig. 8(a), the robot finds the leaving sensor s_3 and switches its state from obstacle to steady. After, the robot broadcasts a message to neighboring steady sensors. Upon receiving the signal strength reports from the neighboring sensors, the robot selects sensor s_3 that has the strongest strength as its guiding sensor and selects sensor s_4 as its reference sensor. Then, the robot adjusts its location and deploys sensor s_{28}. After this, the robot applies the deployment protocol designed for steady state. As shown in Fig. 8(b), the robot deploys sensors from s_{28} up to s_{36}.

When the robot moves to a sensor with an obstacle state, the robot again switches its state from steady to obstacle and treats the sensors in the obstacle state as virtual obstacles. Then, the robot adopts the obstacle surrounding movement policy and moves around the obstacles in a clockwise direction to deploy sensors. The robot deploys sensors around the obstacles until it reaches a leaving sensor. As shown in Fig. 8(b), after the robot deploys sensor s_{36}, it finds that sensor s_{10} is in the obstacle state. The robot switches its state from steady to obstacle. After, the robot treats sensors s_{10}–s_{27}, whose states are obstacle, as virtual obstacles. The robot then deploys sensors around the obstacle until it encounters leaving sensor s_{28}. In total, the robot deploys sensors s_{37}–s_{56} around the virtual obstacles and then switches to steady state. The robot adopts movement and deployment rules designed for steady and obstacle states until the termination condition is satisfied.

C. Layer Information and Energy Conservation

The layer information is important for energy conservation. Another use of layer information is to determine the next movement direction when the robot stays in steady state. When the robot stays in steady state, it moves layer by layer and notifies the deployed sensor with the current layer information. Let the layer information maintained by the robot be k when it encounters the obstacle. The robot switches from steady to obstacle state and deploys m sensors by applying the *obstacle surrounding movement policy* before it encounters the leaving sensor. After this, the robot switches from obstacle to steady state and increases the layer number by one, continuing to deploy sensors in a spiral movement manner. As it deploys m sensors, the robot provides them with the layer information of k, regardless of their physical locations. Since the deployment of sensors near the obstacle highly depends on the obstacle shape, the layer information is useless in helping the robot determine the moving direction. To simplify the maintenance of layer information for the robot staying in the obstacle state, the robot need not change its layer number whenever it switches from steady to obstacle state.

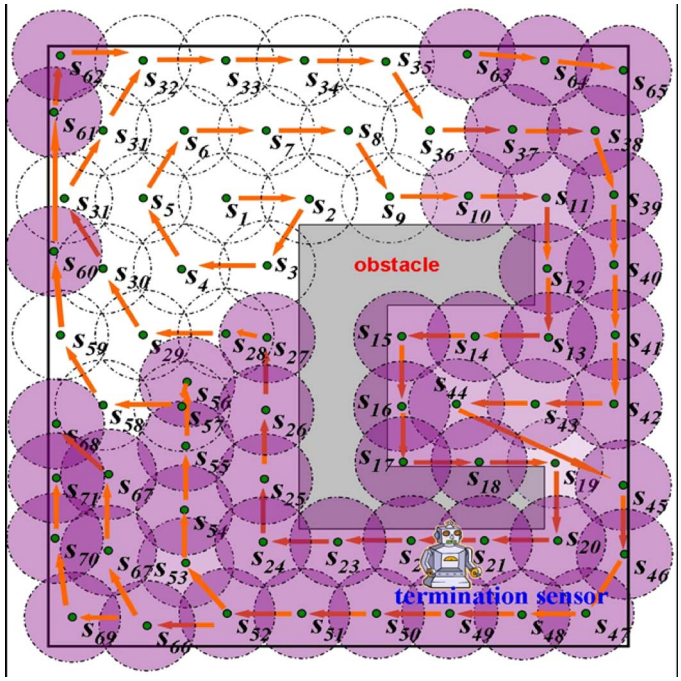


Fig. 12. Execution of *Algorithm Termination*.

To save on energy consumption, only deployed sensors that belong to the largest layer stay in active mode, while other sensors stay in sleep mode for energy conservation. When the robot deploys a sensor, for example, s, with layer k, it broadcasts the layer information to this sensor. Upon receiving the layer information, each sensor compares its maintained layer number with k. In case the maintained layer is smaller than k, it can switch to sleep mode. This technique can work correctly for all deployed sensors, regardless of whether they are deployed in steady or obstacle states.

D. Algorithm Termination

The robot will terminate the deployment process once the monitoring area has full coverage. The robot can then perform other missions such as maintaining network stability.

This section presents how the robot terminates the deployment process. If the robot adopts the movement and deployment rules but does not deploy any sensor for a period of time T, this means that each location was monitored by at least one sensor. The robot selects the nearest sensor to be the terminating sensor. The robot will then keep executing the movement and deployment rules. If the robot encounters a terminating sensor

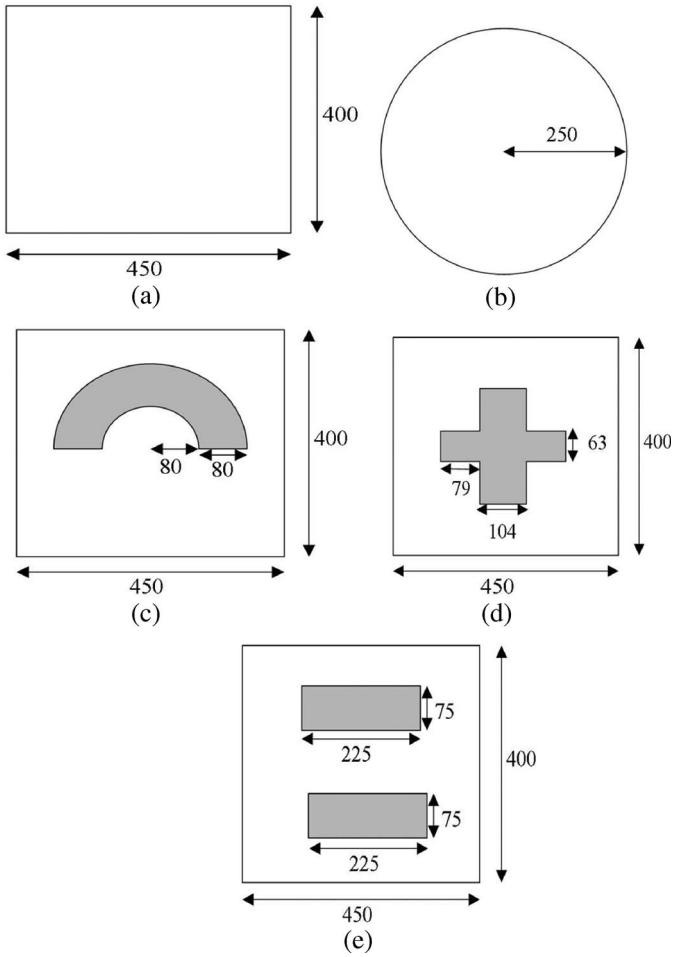


Fig. 13. Considered monitoring areas that might contain obstacles. (a) Scenario 1: Rectangular area. (b) Scenario 2: Circular area. (c) Scenario 3: U-shaped obstacle existing in the rectangular area. (d) Scenario 4: X-shaped obstacle existing in the rectangular area. (e) Scenario 5: Multiple obstacles existing in the rectangular area.

twice but does not deploy any sensor in the monitoring field, the robot terminates the deployment process. As shown in Fig. 12, after a period of time T, the robot does not deploy any sensor on the monitoring field. Next, it selects sensor s_{49} as the terminating sensor. The robot will then keep executing the movement and deployment rules. When the robot encounters sensor s_{49} again, the robot terminates the deployment process.

E. Robot-Deployment Algorithm

The following gives a formal description of the proposed robot-deployment algorithm.

Notations:

s_i: the sensor that its ID $= i,\ i \in Z^+$.
s_g: guiding sensor
s_s: initiator
s_e: leaving sensor
s_r: reference sensor
s_t: terminating sensor
C_R, cr_i: the current round of the robot and s_i.

TABLE I
SIMULATION PARAMETERS

Parameter	Value	Reason
Communication range r_c	20 m	Communication range is greater than $\sqrt{3}\,r_s$
Sensing range r_s	10 m	Typical range observed in many applications
Mobility cost	8.267 J/m	Robomote I[18]
Robot speed	10 m/s	Typical speed observed in many applications
Transmission rate	250 kb/s	Typical transmission rate observed in many applications
Control packet size	50 bytes	Typical control packet size observed in many applications
Packet transmission cost	0.075 J/s	25 mA(TX) at 3V from Berkeley motes
Packet reception cost	0.030 J/s	10 mA(RX) at 3V from Berkeley motes
Idle cost	0.025 J/s	8 mA(processor) at 3V from Berkeley motes
Sleep cost	0.001 J/s	2 μA at 3V from Berkeley motes

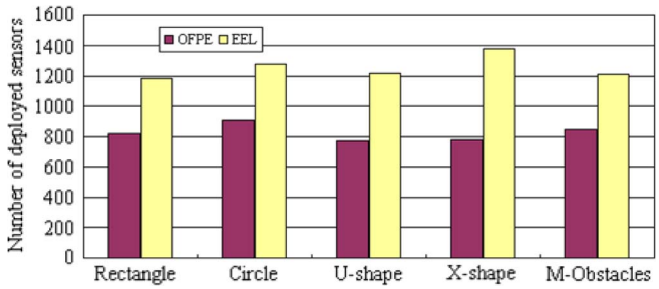


Fig. 14. Average number of deployed sensors in different monitoring areas.

C_D, cd_i: the current direction of the robot and s_i.
C_S, cs_i: the current state of the robot and s_i.
(0) Initialization
  set $d_m = \{0, 5/3\pi, 4/3\pi, \pi, 2/3\pi, 1/3\pi\},\ 1 \le m \le 6$.
  set $C\_R = 0$, $C\_S = steady\_state$,
  Drop a sensor s_1
  If (no obstacle encounters) then
    $C\_S = steady\_state$
  else
    $C\_S = obstacle\_state$
(1) Case 1: $C\_S = steady\_state$
  If (new round begins: $C\_R =$ end or a round) then
    If ($C\_R \neq 0$ and $C\_R \neq 1$) then
      broadcast $sleep\_msg\langle C\_R\rangle$
    produce $D_k = \{d_1^k, d_2^{k-1}, d_3^k, d_4^k, d_5^k, d_6^k\}$ through C_R
    then robot deploys sensors according to D_k
    else ($C\_R \neq$ end or a round)
    send $record\_msg\langle C\_R, C\_D, C\_S\rangle$ to s_{i-1}
    select s_g and s_r by broadcasting $search\_beacon$
    robot moves in C_D direction listed in D_k for a distance of $\sqrt{3}r_s$ and deploy s_i
    If (no sensors deployed during period time T) then
    send $termination\_msg$ to s_g as s_t
  If (robot encounters s_t again without deploying any sensor)

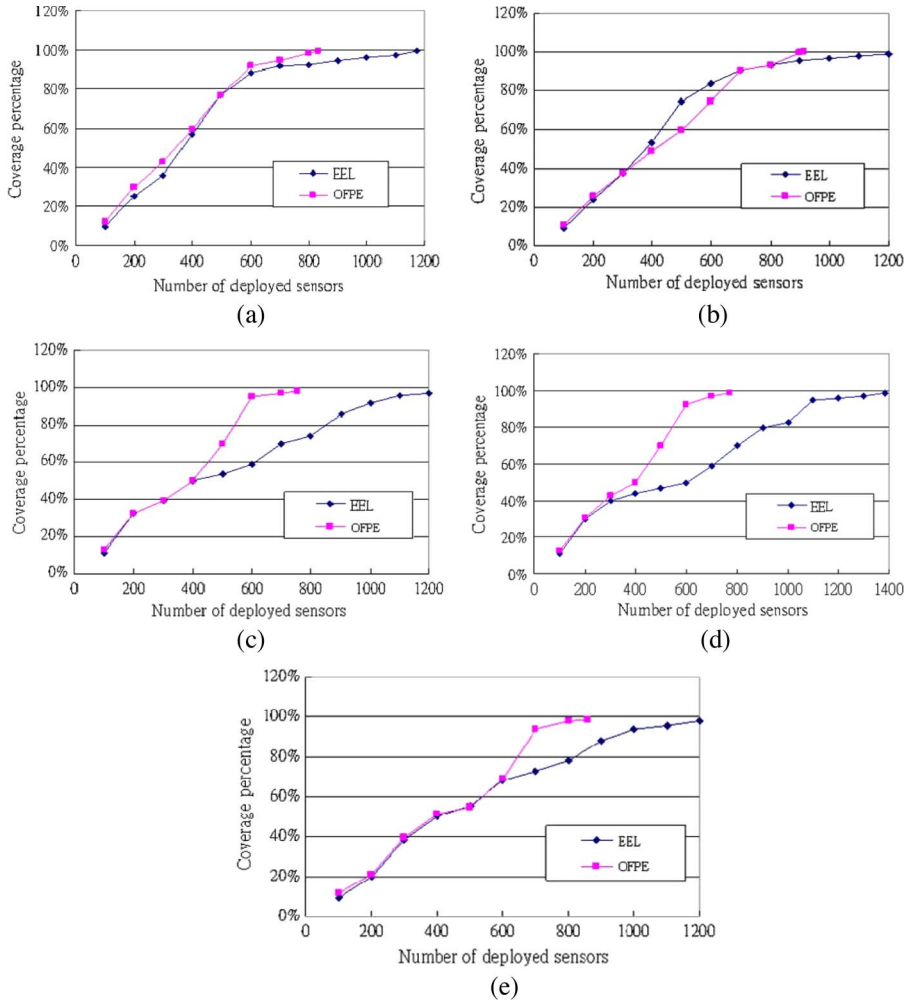


Fig. 15. Relationship between the number of sensors and the coverage percentages in different scenarios. (a) Scenario 1: Rectangular area. (b) Scenario 2: Circular area. (c) Scenario 3: U-shaped obstacle. (d) Scenario 4: X-shaped obstacle. (e) Scenario 5: Multiple obstacles.

then
algorithm termination
If (obstacle encounters obstacle or $cs_{\mathrm{MRDS}} = obstacle\_state$)
then
$C\_S = obstacle\_state$
send set_msg to s_g to play the role of s_s
broadcast $obstacle\_msg$ and the sensors near obstacle record themselves as s_e

(2) Case 2:$\boldsymbol{C\_S} = \boldsymbol{o}bstacle\_state$
Generate the local information of the obstacle
If (robot encounters s_e)
$C\_S = steady\_state$
broadcast ref_msg to find s_g and s_r and then deploy s_i at expected location
If (robot locates in sensing range of some sensor s_k) then
move clockwise around the obstacle without deployment of sensor
else
deploy a sensor and move clockwise around the obstacle
If (no sensors deployed during period time T) then
send $termination\_msg$ to s_g as s_t
If (robot encounters s_t without deploying any sensor) then
algorithm termination

V. Performance Study

This section compares the performance of the proposed obstacle-free and power-efficient deployment algorithm (OFPE) with another related work, *Efficient Exploration without Localization* (EEL) [14].

1) Simulation Environment: The performance study uses five different shapes of monitoring fields where obstacles may exist, as shown in Fig. 13. In each simulation, the initial location of the robot is randomly determined in the monitoring field. The cardinal direction that the robot first faces is east. The other cardinal directions are separated clockwise per $\pi/3(d_2 = 5/3\pi, d_3 = 4/3\pi, d_4 = \pi, d_5 = 2/3\pi, d_6 = \pi/3)$. All sensors have the same capability. The communication and sensing ranges are r_c and r_s, respectively, where $r_c \geq \sqrt{3}r_s$. Table I details the values set for the simulation parameters.

For fairness, the predefined distance in [14] is equal to the best deploying distance $\sqrt{3}r_s$. The performance is evaluated in terms of the average number of deployed sensors, the average

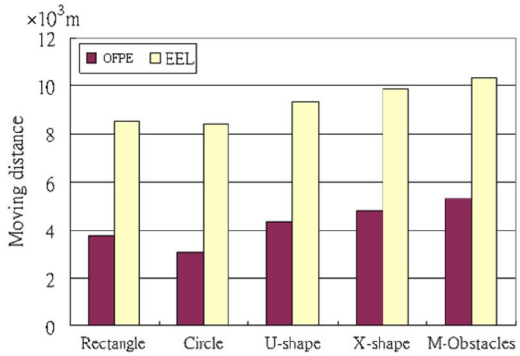


Fig. 16. Average moving distance in various scenarios.

moving distance, the power consumption, and the number of check circles. The simulation results are obtained from the average of 20 runs.

2) Average Number of Deployed Sensors: Fig. 14 shows the number of sensors used in different monitoring fields. OFPE takes into account the obstacles so that there would be no coverage holes and the overlapping area of the sensing range would be minimal. As a result, OFPE outperforms EEL by at least 28.64% and at best 44.41%.

Fig. 15 shows the relationship between the number of sensors and the coverage percentages in several different monitoring areas. In particular, Fig. 15(a) and (b) shows the coverage percentage without the existence of obstacles in two monitoring areas. Without the impact of obstacles on deployment, the coverage percentages of EEL and OFPE are similar. However, EEL deploys more sensors than OFPE in achieving a similar coverage percentage. Fig. 15(c), (d), and (e) shows the performance in terms of coverage percentage when U- and X-shaped obstacles exist in the monitoring areas. With the impacts of obstacles on deployment, the OFPE outperforms EEL in coverage percentage. The proposed OFPE can overcome obstacles and efficiently reduce the impact of obstacles in deployment efficiency. However, in case the number of obstacles increases, these obstacles would significantly impact OFPE and EEL. This phenomenon can be found in Fig. 15(e).

3) Average Moving Distance: The movement policy and the shape of the monitoring region determine the moving distance for the robot when deploying sensors to achieve full coverage. Fig. 16 shows the average moving distance by applying OFPE and EEL in a rectangular or circular region. The OFPE requires smaller distances than EEL because the robot movement of EEL is only guided by a single deployed sensor, so this deployment easily results in many small holes, requiring the robot to move to further deploy sensors to achieve full coverage. Another limitation is that EEL did not take obstacles into consideration in developing its robot-deployment algorithm. Different shapes of obstacles, including the U, X, and M shapes, are randomly generated in the rectangular and circular monitoring regions. When the robot encounters obstacles, EEL results in many holes and thus requires additional movements to achieve full coverage.

4) Average Power Consumption: Fig. 17(a) shows the average power consumption of robot movement, which is proportional to the average moving distance. The OFPE outperforms EEL by a 48.67%–63.13% improvement in terms of power consumption. In addition to the factor of robot movement, the number of transmitting, receiving, and idle listening packets created during the deployment process also impacts the power consumption of the robot and the deployed sensors. Fig. 17(b) shows the required average power consumption of the robot and the deployed sensors from packet transmissions and idle listening. By applying OFPE, most deployed sensors stay in sleep mode. As a result, OFPE outperforms EEL by a 64.54%–82.09% improvement in power consumption.

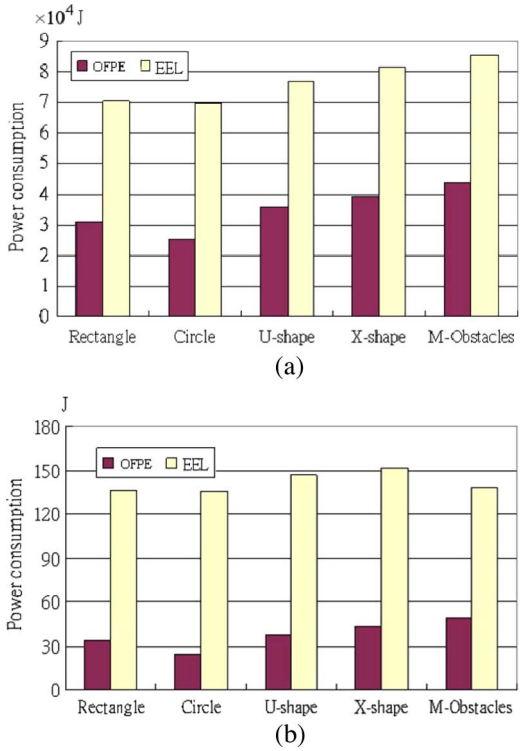


Fig. 17. Power consumption of robot movements and communications. (a) Power consumption in robot movements. (b) Power consumption in communications.

VI. Conclusion and Future Work

This paper proposes a robot-deployment algorithm that efficiently handles the problem of obstacles and likely achieves the purposes of power conservation and full coverage with the deployment of fewer sensors. The proposed robot-deployment algorithm consists of steady and obstacle states. In steady state, spiral movement and node placement policies are proposed to achieve energy conservation and full coverage using fewer deployed sensors. When the robot encounters an obstacle, its algorithm switches to obstacle state wherein the robot adopts the obstacle surrounding movement policy to move and deploy sensors, reducing the impacts of obstacles on deployment. Furthermore, the robot treats sensors that are deployed in obstacle state as virtual obstacles and deploys sensors layer by layer according to the spiral movement policy. Simulation results show that the proposed algorithm significantly reduces the number of deployed sensors and improves the resistance

to obstacles. Future work can consider a team of robots that cooperatively execute the same deployment task in a distributed manner. Furthermore, a WSN is typically expected to work for a long period of time. Sensors in some regions might fail because of energy exhaustion. Another possible work in the future could focus on developing an efficient robot redeployment algorithm that maintains full coverage by deploying fewer sensors within a reasonable time duration.

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] M. Krysander and E. Frisk, "Sensor placement for fault diagnosis," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 6, pp. 1398–1410, Nov. 2008.

[3] A. Chen, T. H. Lai, and D. Xuan, "Measuring and guaranteeing quality of barrier-coverage in wireless sensor networks," in *Proc. 9th ACM Int. Symp. Mobile Ad Hoc Netw. Comput. (MobiHoc)*, Hong Kong, Jun. 2008, pp. 421–430.

[4] J. M. Kay and J. Frolik, "An expedient wireless sensor automaton with system scalability and efficiency benefits," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 6, pp. 1198–1209, Nov. 2008.

[5] R. Kleinberg, "Geographic routing using hyperbolic space," in *Proc. 26th Annu. Joint Conf. IEEE Comput. Commun. Societies (INFOCOM)*, Anchorage, AK, May 2007, pp. 1902–1909.

[6] Z. Gao, T. Breikin, and H. Wang, "Reliable observer-based control against sensor failures for systems with time delays in both state and input," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 5, pp. 1018–1029, Sep. 2008.

[7] J. M. Glasgow, G. Thomas, E. Pudenz, N. Cabrol, D. Wettergreen, and P. Coppin, "Optimizing information value: Improving rover sensor data collection," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 3, pp. 593–604, May 2008.

[8] C. Y. Chang and H. R. Chang, "Energy-aware node placement, topology control and MAC scheduling for wireless sensor networks," *Comput. Netw.*, vol. 52, no. 11, pp. 2189–2204, Aug. 2008.

[9] W. Li and C. G. Cassandras, "A minimum-power wireless sensor network self-deployment scheme," in *Proc. Annu. IEEE WCNC*, New Orleans, LA, Mar. 2005, vol. 3, pp. 1897–1902.

[10] G. L. Wang, G. H. Cao, and T. LaPorta, "Movement-assisted sensor deployment," in *Proc. 23rd Annu. Joint Conf. IEEE Comput. Commun. Societies (INFOCOM)*, Hong Kong, Mar. 2004, vol. 4, pp. 2469–2479.

[11] G. L. Wang, G. H. Cao, T. LaPorta, and W. S. Zhang, "Sensor relocation in mobile sensor networks," in *Proc. 24rd Annu. Joint Conf. IEEE Comput. Commun. Societies (INFOCOM)*, Miami, FL, Mar. 2005, pp. 2302–2312.

[12] A. Sekhar, B. S. Manoj, and C. S. R. Murthy, "Dynamic coverage maintenance algorithms for sensor networks with limited mobility," in *Proc. 3rd IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Kauai Island, HI, Mar. 2005, pp. 51–60.

[13] 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.

[14] M. A. Batalin and G. S. Sukhatme, "Efficient exploration without localization," in *Proc. 2003 IEEE Int. Conf. Robot. Autom. (ICRA)*, Taipei, Taiwan, Sep. 2003, vol. 2, pp. 2714–2719.

[15] M. A. Batalin and G. S. Sukhatme, "Coverage, exploration and deployment by a mobile robot and communication network," in *Proc. Int. Workshop IPSN*, Apr. 2003, pp. 376–391.

[16] M. A. Batalin and G. S. Sukhatme, "Sensor coverage using mobile robots and stationary nodes," in *Proc. SPIE Conf. Scalability Traffic Control IP Netw. II (Disaster Recovery Netw.)*, Boston, MA, Aug. 2002, pp. 269–276.

[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. 1st IEEE Int. Conf. Wireless Internet (WICON)*, Visegrad-Budapest, Hungary, Jul. 2005, pp. 114–121.

[18] G. T. Sibley, M. H. Rahimi, and G. S. Sukhatme, "Robomote: A tiny mobile robot platform for large-scale sensor networks," in *Proc. IEEE ICRA*, Washington, DC, May 2002, pp. 1143–1148.

Chih-Yung Chang received the Ph.D. degree in computer science and information engineering from National Central University, Taipei, Taiwan, in 1995.

He was with the faculty of the Department of Computer and Information Science, Aletheia University, Taipei, as an Assistant Professor in 1997, where he was the Chair from August 2000 to July 2002. Since August 2002, he has been with the Department of Computer Science and Information Engineering, Tamkang University, Tamsui, Taiwan, where he was an Associate Professor and is currently a Full Professor. He was an Associate Guest Editor of the *Journal of Information Science and Engineering* (2008), the *Journal of Internet Technology* (2004 and 2008), and the *Journal of Mobile Multimedia* (2005) and was a member of the Editorial Board of the *Tamsui Oxford Journal of Mathematical Sciences* (2001–2008) and the *Journal of Information Technology and Applications* (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 the IEEE Computer and Communication Societies and the Institute of Electronics, Information and Communication Engineers Society.

Jang-Ping Sheu (SM'98) received the B.S. degree in computer science from Tamkang University, Tamsui, Taiwan, in 1981 and the M.S. and Ph.D. degrees in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 1983 and 1987, respectively.

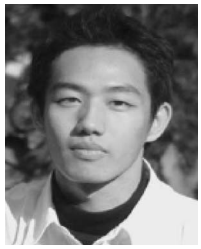
He was the Chair of the Department of Computer Science and Information Engineering, National Central University, Jhongli City, Taiwan, from 1997 to 1999, where he was the Director of the Computer Center from 2003 to 2006. He is currently a Chair Professor of the Department of Computer Science and Information Engineering, National Tsing Hua University. He is an Associate Editor of the *International Journal of Ad Hoc and Ubiquitous Computing* and the *International Journal of Sensor Networks*. He was an Associate Editor for the *Journal of the Chinese Institute of Electrical Engineering*, the *Journal of Information Science and Engineering*, the *Journal of the Chinese Institute of Engineers*, and the *Journal of Internet Technology*. His current research interests include wireless communications and mobile computing.

Dr. Sheu is a member of the Association for Computing Machinery and the Phi Tau Phi Society. He is an Associate Editor for the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS. He was the recipient of the Distinguished Research Awards from the National Science Council of the Republic of China in 1993–1994, 1995–1996, and 1997–1998, the Distinguished Engineering Professor Award from the Chinese Institute of Engineers in 2003, the certificate of Distinguished Professorship from the National Central University in 2005, and the K.-T. Li Research Breakthrough Award from the Institute of Information and Computing Machinery in 2007.

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

His research domains include wireless sensor networks, ad hoc wireless networks, mobile/wireless computing, and WiMAX.

Mr. Chen won many scholarships in Taiwan and participated in many wireless sensor networking projects.



Sheng-Wen Chang received the B.S. degree in computer science and information engineering from Tamkang University, Tamsui, Taiwan, in 2004, where he has been working toward the Ph.D. degree since 2005 in the Department of Computer Science and Information Engineering.

He has published extensively in the wireless networking area. His research interests are WiMAX, wireless sensor networks, Bluetooth radio networks, wireless mesh networks, and ad hoc wireless networks, concerning both theoretic and algorithm design.

Mr. Chang is a student member of the IEEE Computer and Communication Societies and the Institute of Electronics, Information and Communication Engineers Society. He has won many scholarships in Taiwan and participated in many Bluetooth and wireless sensor networking projects.