WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2013; **13**:377–392
Published online 1 February 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/wcm.1106

RESEARCH ARTICLE

An energy-efficient hole-healing mechanism for wireless sensor networks with obstacles

Chih-Yung Chang[1]*, Chih-Yu Lin[1], Gwo-Jong Yu[2] and Chin-Hwa Kuo[1]

[1] Department of Computer Science and Information Engineering, Tamkang University, Taipei 251, Taiwan
[2] Department of Computer and Information Science, Aletheia University, Taipei 251, Taiwan

ABSTRACT

In wireless sensor networks (WSNs), coverage of the monitoring area represents the surveillance quality. Since sensor nodes are battery powered and placed outdoor, there will be failures due to energy exhaustion or environmental influence, resulting in coverage-loss. In literature, a number of studies developed robot repairing algorithms that aim at maintaining full coverage. However, they did not consider the time constraint for network maintenance. Furthermore, they did not consider the existence of obstacles and the constraint of limited energy of the robot. This paper presents a novel tracking mechanism and robot repairing algorithm for maintaining the coverage quality of the given WSN. Without support of location information, the tracking mechanism leaves robot's footmark on sensors so that they can learn better routes for sending repairing requests to the robot. Upon receiving several repairing request messages, the robot applies the proposed repairing algorithm to establish an efficient route that passes through all failure regions with low overhead in terms of the required time and the power consumption. In addition, the proposed repairing algorithm also considers the remaining energy of the robot so that the robot can move back to home for recharging energy and overcome the unpredicted obstacles. Performance results reveal that the developed protocol can efficiently maintain the coverage quality while the required time and energy consumption are significantly reduced. Copyright © 2011 John Wiley & Sons, Ltd.

KEYWORDS

deployment; coverage quality; repair; sensor network; robot

***Correspondence**

Chih-Yung Chang, Department of Computer Science and Information Engineering, Tamkang University, Taipei 251, Taiwan.
E-mail: cychang@mail.tku.edu.tw

1. INTRODUCTION

Wireless sensor networks (WSNs) compose of many sensor nodes embedded with simple processor, few memory, tiny sensing material, and energy-limited battery. WSNs have been widely applied on many applications, including environmental monitoring, tracking, precision agriculture, military surveillance, smart homes, and so on [1–6]. The accuracy of sensing information depends on the coverage quality which highly depends on the deployment and redeployment mechanisms.

Random deployment is the simplest way to deploy a huge number of sensor nodes in a specific region but may cause the unbalanced density. Sensors in a dense deployment region may overlap too much sensing range and therefore increase the hardware cost. However, a spare deployment region may raise the coverage-hole problem. As a result, the random deployment presents problems of high hardware cost or coverage hole.

Other than the random deployment, an alternative is the robot deployment where the robot deploys sensors stepwise in a monitoring area. The robot deployment can achieve the full coverage purpose by using minimal number of sensor nodes. In literature, previous studies [7–11] adopt robot to deploy sensors in a dangerous region that is unsuitable for human deployment. Some researches [7–14] exploit the robot to execute important tasks such as sensor deployment, patrol, or hole repair. In literature, researches [7,8] assume that the robot equipped with a compass which makes robot aware of its moving direction without location information. The robot deploys sensors according to the predefined direction priority. Although the proposed robot deployment scheme is likely to achieve the purpose of full coverage and network connectivity in the environment without existing obstacles, it may cause too much sensing redundancy and cannot guarantee full coverage if the robot encounters obstacles. Furthermore, the developed robot movement policy did not take into account the robot's energy constraint.

Copyright © 2011 John Wiley & Sons, Ltd.

The robot may exhaust its energy during the execution of deployment task.

Research [10] proposed an obstacle-free snake-like robot deployment protocol, called OFRD, which aims at deploying minimal number of sensor nodes to achieve full sensing coverage even though there are unpredicted obstacles. However, OFRD did not discuss how the robot repairs the coverage-loss in WSN. Since sensor nodes are battery powered and deployed outdoor, they might fail due to energy exhaustion or environmental influence, and hence result in the WSN coverage-loss. To maintain the coverage quality, the redeployment in the failure regions is necessary. Pervious work [8,11] developed robot deployment and redeployment algorithms without the need of location information. Each deployed sensor counts the time interval that the robot does not visit for each direction when executing the deployment task. After completing the deployment task, the robot executes patrolling task. The deployed sensors guide the robot's movement by suggesting a suitable direction that has maximal time interval for executing the patrolling and redeployment tasks. The article also employs a *Home* algorithm that enables the robot going home for the energy recharge. However, the robot did not leave its trajectory on the sensors and thus the sensors that are nearby the failure regions cannot timely notify the robot for executing the repairing task. Hence the performance of repairing task is inefficient in terms of the time and energy consumption. In addition, obstacles are not considered in the robot repairing phase, making the robot unable to guarantee that the remaining power is enough to move to home. Therefore, it is important to develop an efficient robot repairing mechanism that schedules an efficient moving path for repairing the failure regions with the consideration of robot's energy constraint.

At a glace, the repairing path construction problem is similar to the well known Traveling Salesman Problem (TSP) [15–17]. However, they are quite different because that the remaining energy of the robot is not taken into consideration in TSP. The energy of robot will be exhausted while executing the repair task. How to efficiently arrange the home location in the repairing path even though the obstacle exists in the environment is a big challenge.

This paper presents novel tracking mechanism and robot repairing algorithm. Based on the robot deployment algorithm proposed in Refs. [7–11], the tracking mechanism leaves the movement trajectory information of the robot on each deployed sensor. According to this information, sensors nearby the failure region can efficiently learn how to rapidly deliver the request message to the robot in a distributed manner. Then the proposed robot repairing algorithm constructs the shortest path for repairing multiple failure regions. As a result, the proposed repairing algorithm efficiently takes minimal time and consumes minimal energy to recover the failure regions. Unlike TSP, the path construction for maintaining the coverage quality also considers the remaining energy of the robot so that the robot can be back to home for recharging energy and overcome the unpredicted obstacles.

The remaining parts of this paper are organized as follows. Section 2 introduces the related work and basic concepts of the proposed algorithms. Section 3 illustrates the network environment and problem formulation of this paper. Section 4 depicts the tracking mechanism. The Robot Repairing algorithm is proposed in Section 5. Section 6 examines the performance of the proposed algorithms while Section 7 concludes this work.

2. RELATED WORK AND BASIC CONCEPTS

This section reviews the related work of robot deployment, patrolling, and repairing mechanisms and then illustrates the basic concept of the proposed robot repairing algorithm.

2.1. Related work

In literature, several deployment mechanisms have been proposed for the robot to stepwise deploy static sensors in a specific region. Previous research [8,11] assumes that the robot is equipped with a compass and is able to detect obstacle. To guide the robot's movement for executing the patrolling and repairing tasks, each deployed sensor maintains the time interval that the robot did not visited for each direction of south, east, north, and west. When the robot intends to make a decision of patrolling movement direction, it communicates with the closest deployed sensor, and then moves toward the direction that has the largest value of time interval maintained at that sensor. Although the coverage quality can be maintained by robot's patrolling, the occurrence of sensor failures might not be related to the time interval. For example, frequent events occurred at a region might cause the sensors of that region to exhaust their energy. As a result, the failure region should passively wait for the robot's visiting. The robot did not leave the trajectory on the deployed sensors and hence those sensors that are nearby the failure regions are unable to send the repairing request to the robot. Consequently, the robot might visit the failure regions after a long period of time. The patrolling mechanism proposed in Ref. [8] also employs a Home algorithm that enables the robot going home for the energy recharge. However, the remaining energy of the robot and the existence of obstacles are not taken into consideration. Consequently, the robot might exhaust its energy during the execution of repairing task.

Research [10] proposes an OFRD mechanism that deploys the monitoring region with near-minimal number of sensors and likely achieves the full coverage purpose. The deployment algorithm deploys minimal number of sensors in the environment with and without obstacles, respectively. However, since sensor nodes are battery powered and are deployed in the outdoor environment, they might fail due to energy exhaustion or environmental influence, and hence the WSN would suffer from coverage-loss.

However, previous studies [7–11] did not consider the network maintenance problem.

To address the abovementioned drawbacks, we propose a novel tracking mechanism and a robot repairing algorithm. With the extension of the existing robot deployment algorithms [7–11], the tracking mechanism is proposed to enable the robot leaving the trajectory information on each deployed sensor. According to this information, sensors nearby the failure region can efficiently notify the robot for repairing. When the robot has collected several request notifications during a given time duration, it applies the proposed repairing algorithm to construct a repairing path. The proposed repairing algorithm considers the existence of unpredicted obstacles and the constraint of robot's remaining energy. As a result, the proposed repairing algorithm efficiently takes minimal time and consumes minimal energy to recover the failure regions.

2.2. Basic concepts of the tracking robot and robot repairing mechanisms

Since sensor nodes are battery powered and placed at the outdoor environment, they might fail due to energy exhaustion or environmental influence, resulting in the WSN coverage-loss. To maintain the monitoring quality, a robot that carries new sensor nodes is responsible for repairing the failure region. Let *request initiator* denote the sensor that is nearby the failure region and intends to initiate the repairing request to the robot. To help the request initiator send request message to the robot, the movement trajectory of the robot should be transparent to all sensor nodes. Therefore, the first task of this article is to extend the existing robot deployment algorithms [7–11] so that the robot can leave footmark on each deployed sensor. According to the footmarks, the request initiators are able to deliver their request messages to the robot. In addition, the route for delivering the message might be rugged and inefficient. The second goal of this paper aims to help all sensors learn better routes from themselves to the robot. To achieve this goal, this paper proposes an X-correction mechanism for correcting the footmarks when the robot executes the network patrolling or repairing tasks. By applying the proposed X-correction mechanism, the footmarks automatically guide the request packet to be forwarded along a better route.

As shown in Figure 1, the solid path denotes the trajectory of robot. When the robot is located at location B, a request initiator located at location A intends to send the repairing request message to the robot. By applying the proposed X-correction mechanism, the request packet will be delivered from location A to location B along the path marked by the dotted line. The length of the learned path (dotted line) is shorter than that of the trajectory path (solid line).

Upon receiving several request messages from different initiators within a certain time period, the robot will construct a path that passes through all failure regions for repairing the WSN. The next goal of this paper is to develop a path construction algorithm that constructs

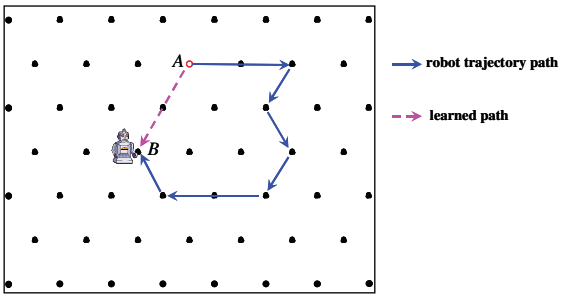

Figure 1. The proposed X-correction mechanism help sensor learn a better route from itself to the robot.

a shortest path for reducing the recovery time and saving the robot's energy consumption. Figure 2 compares the constructed paths by applying the greedy algorithm and the proposed repairing algorithm. In Figure 2, nodes A, B, C, D, and E denote the failure regions. The greedy algorithm greedily selects the nearest failure region as the next visiting target and constructs an edge from the current location to the selected target until the path passing through all failure locations. Therefore the robot will move along the path $O \rightarrow A \rightarrow C \rightarrow B \rightarrow D \rightarrow E$, which is marked by the dotted line as shown in Figure 2. The developed repairing algorithm would establish a better path $O \rightarrow C \rightarrow A \rightarrow B \rightarrow D \rightarrow E$, which is marked by solid line. In comparison, the length of path constructed by the proposed repairing algorithm is better than that of greedy algorithm due to the fact of $\overline{OA} + \overline{OB} > \overline{AB}$.

Since the robot has the energy constraint, constructing a path for repairing the WSN should take into consideration the Home where the robot can recharge its energy. Figure 3a gives an example of the route construction with considering the home problem. Assume nodes A, B, C, D, and E are the failure regions. We assume that the robot exhausts its energy during the movement from node B to node D. A straightforward solution is to firstly construct a shortest path without considering the home location, then insert the home location to the constructed shortest path. However, the resultant path might not be efficient since the home location might be far away from the failure regions visited before and after the home location. For example, inserting the home location between failure regions B and D will create a path $C \rightarrow A \rightarrow B \rightarrow$ Home $\rightarrow D \rightarrow E$, as shown in Figure 3a, which results in two inefficient segments $B \rightarrow$ Home and

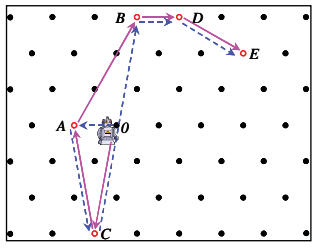

Figure 2. The proposed repairing algorithm constructs a shortest path marked by the solid line.

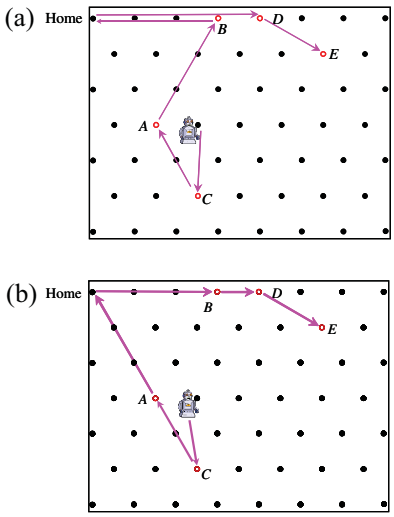

Figure 3. The developed repairing algorithm constructs an efficient path that prevents the robot from energy exhaustion. (a) Inserting home location in the shortest route is not a feasible solution for preventing the robot from energy exhaustion. (b) The path constructed by applying the proposed repairing mechanism.

Home $\rightarrow D$. The proposed repairing algorithm enables the robot to repair multiple failure regions rapidly and arranges the home location in proper order so that the robot will not exhaust its energy during the execution of repairing task. As shown in Figure 3b, the robot considers the remaining energy and firstly repairs region A. Then, the robot goes back to home region for energy supply. After that, the robot repairs the failure regions B, D, E, C in order. Therefore, the robot not only recharges its energy, but also rapidly repairs all failure regions.

Another important feature of the proposed repairing algorithm is that it takes the unpredicted obstacles into consideration. The existence of unpredicted obstacles will raise the problem that the expected path is shorter than the actual path since the constructed path may be blocked by the obstacle. The unknown obstacles might raise the problem that the robot cannot make sure that its remaining energy is enough to traverse the actual path. To cope with this problem, the developed repairing algorithm initiates a probe packet to estimate the impact of obstacle on the energy consumption. As shown in Figure 4, nodes A, B, C, D, E, and F denote the failure regions and the path $O \rightarrow A \rightarrow C \rightarrow F \rightarrow E \rightarrow D \rightarrow B$ is the shortest routing path passing through all failure regions. The robot initiates a probe packet to evaluate the length of actual path and hence constructs a repairing path according to the remaining energy. In case that the remaining energy is not enough to traverse the path, the home location will be visited by robot before the robot exhausts its energy. As shown in Figure 4, according to the length estimation information, the robot inserts home location between regions E and F and constructs the path which is marked by the solid line. Consequently, the robot can complete the repairing task even

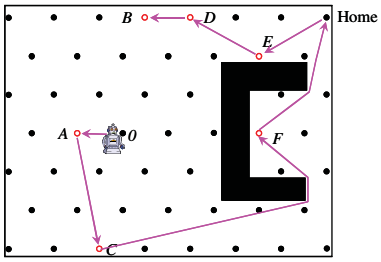

Figure 4. The proposed repairing algorithm constructs an efficient path even though it encounters unpredicted obstacles.

though the obstacles block paths from region C to region F and from region F to region E.

In summary, this paper designs a tracking mechanism and an obstacle-free robot repairing algorithm for maintaining the monitoring quality of a WSN. The tracking mechanism helps all sensors learn better routes to deliver the request messages to the robot while the repairing algorithm takes the remaining energy of robot and obstacles into consideration and constructs an efficient path that passes through all failure regions even though there exist unpredicted obstacles.

3. NETWORK ENVIRONMENT AND PROBLEM FORMULATION

3.1. Network environment

This paper considers a single robot that carries several sensor nodes and is equipped with a Global Positioning System (GPS) through which the robot is aware of its location and moving direction. In literature, a lot of robot deployment mechanisms [7–11] have been proposed for deploying a given monitoring region. These mechanisms use the robot to deploy the sensors in a way that every sensor has exactly six neighbors so that the number of deployed sensors is minimal while the whole monitoring region can be full covered. In this paper, we assume that a number of sensors have been deployed in the monitoring region by applying any existing robot deployment mechanisms proposed in Refs. [7–11]. Since sensor node is battery powered, the WSN might have coverage loss after working for a long time. The robot stays in the monitoring region for the task of coverage maintenance. The robot can perform other tasks such as patrolling and data collection. Each deployed sensor is aware of its own location. However, the current location of robot is not known by sensors since the robot might change its location with time for executing its task. The sensors that detect a coverage hole will immediately initiate a hole-healing request and send requests to the robot for healing the holes. These sensors are called *request initiators*. The robot will perform the hole-healing task when several requests have received or a given time duration has been elapsed after the first request has been received. Herein, we assume that the failure of sensors at any moment will not

cause the network partition. With this assumption, the network is connected so that the initiator can detect the failure occurrence of any neighboring sensor and notify the robot. We also assume that the robot needs at most one energy recharge for executing the deployment and hole-healing tasks.

3.2. Problem formulation

This article copes with the following two problems. First, we aim to design a tracking robot mechanism based on the existing deployment algorithms [7–11]. The designed tracking mechanism leaves the robot's trajectory on the deployed sensors to enable the request initiators efficiently tracking where the robot is and sending the hole-healing request to the robot for maintaining full coverage. Another goal of this work is to design a robot repairing algorithm for constructing an efficient path which takes into consideration the existence of obstacles and the constraint of limited energy of the robot.

The following presents the problem formulation of this work. Let S denote the finite set of sensors deployed in a two-dimensional monitoring region A. As soon as the request initiator detects the failure region, it intends to send a repairing request to the robot. Let $I=\{s_1, s_2, \ldots, s_m\}$ denote the set of m request initiators and their representative coordinates are (x_i, y_i), for $1 \le i \le m$. We assume that the Home region H is located at (x_H, y_H). The obstacle might stay at any location of A and the location is unknown. Let $d_{i,j}$ denote the distance between two request initiators s_i and s_j. We have $d_{i,j} = \sqrt{(x_i-x_j)^2 + (y_i-y_j)^2}$ where $i \neq j$. A request initiator s_h sends the repairing request packet along path P_h which contains n_h forwarders $s_{h,1}, s_{h,2}, \ldots, s_{h,n_h}$. That is $P_h = \langle S_{h,1}, S_{h,2}, \ldots, S_{h,n_h} \rangle$. Upon receiving m failure notifications, the robot will establish a path R which passes through m request initiators $s_1, s_2, \ldots,$ and s_m. That is $R = \langle R_1, R_2, \ldots, R_{m+1} \rangle$ where $\forall R_i \in I \cup \{Home\}$ for $1 \le i \le m+1$ and $R_i \neq R_j$ for $i \neq j$. Let the remaining energy of the robot at the moment of receiving requests I be E_{left}.

The scope of this work mainly covers two issues. One is how the request initiators in set I efficiently send the repairing requests to the robot. The other issue is how the robot efficiently moves to repair the failure regions along the construct path R. The total energy cost for the m request initiators notifying the robot about their failure regions is formulated as Equation (1).

$$Cost_{notification} = \sum_{h=1}^{m} \sum_{1 \le i < n_h} \left(E_t \times d_{i,i+1}{}^{\alpha} \right) \tag{1}$$

where E_t denotes the energy consumption for transmitting a repairing request packet to the robot for a unit distance and α is the path loss exponential whose value ranges from 2 to 5.

In addition, the total energy cost for robot's movement along the constructed path R is given by Equation (2):

$$Cost_{movement} = \sum_{1 \le i < m+1} (E_{mov} \times d_{i,i+1}) \tag{2}$$

where E_{mov} denotes the energy consumption for the robot moving for a unit distance.

The object function of this work is given in Expression (3).

$$\text{Minimize}\{Cost_{notification} + Cost_{movement}\}$$

subject to the *power constraint*:

$$E_{left} > \max\{E_{mov} \times d_{i,i+1}, E_{mov} \times d_{i,H}\} \quad \text{for all } R_i \in R,\ 1 \le i \le m+1 \tag{3}$$

4. TRACKING ROBOT MECHANISM

This section presents the tracking mechanism which helps each sensor learn a better route from itself to the robot. The description of tracking mechanism consists of deployment phase, patrolling phase and hole-healing phase as presented below.

4.1. Deployment phase

The basic idea behind the proposed tracking robot mechanism is that the robot leaves footmarks on the deployed sensors when it executes the deployment task. Two types of information are set up on each deployed sensor by the robot. The first type is the location information which is given by the robot since the robot is equipped with a GPS. Each sensor should be aware of its location since the information is important in the WSNs. Many existing studies [18–20] have proposed localization mechanisms to provide each sensor with location information. Using robot to provide each deployed sensor with physical location is simplest task. Figure 5 shows the location information of the deployed sensors. Without loss of generality, this paper uses a regular-deployed WSN to illustrate the proposed tracking mechanisms since the WSN shown in Figure 5 has been proved to be full covered by the minimal number of sensors

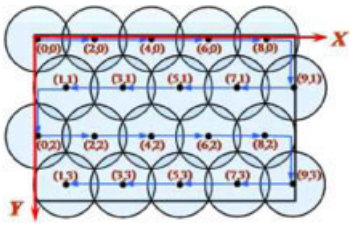

Figure 5. The robot provides each sensor with location information during the execution of deployment phase.

[10]. The proposed mechanism can be treated as an extension of existing robot deployment algorithms proposed in Refs. [7–11].

In addition to the location information, each sensor should additionally maintain the robot's footmark, called *Counter Value*, which provides each sensor with tracking information and helps itself learn a better route to the robot. The counter value is represented as ⟨*Broadcast ID*, *Hops*⟩ where the *Broadcast ID* field maintains the number of broadcasts received by this sensor while the *Hops* field maintains the movement trajectory of the robot. The initial value of the Broadcast ID is 0. Whenever the robot deploys a sensor during the execution of the network deployment task or visits a deployed sensor during the execution of hole-healing task, the value of *Hops* is increased by one. The Hops value will be maintained by the deployed or visited sensors. As a result, in case that the Broadcast ID is 0, a sensor with a larger Hops value represents that the robot is closer to that sensor node. Each sensor should maintain its own counter value and the ID of the neighboring sensor that has the largest counter value.

Each request initiator will utilize the counter value to trace the current robot location and send the repairing request to the robot. The repairing request packet consists of the location information of the failure sensor and the ID of neighbor sensor node which has the largest counter value. Upon receiving the repairing request packet, the node with the ID defined in repairing packet will be responsible for forwarding the packet. It simply forwards the request packet to the neighbor that has the largest counter value. Since a sensor closer to the robot maintains a larger Hops value, the request packet will be delivered to the robot stepwise. The robot will wait for a given time period to receive several request packets from different request initiators. Then the robot applies the proposed repairing algorithm to construct a repairing path that passes through all failure regions and repairs the failure regions along the constructed path.

4.2. Patrolling phase

After the robot finishes the deployment task, it starts executing the patrolling phase. The robot will switch from patrolling phase to the hole-healing phase if it receives any hole-healing request from any sensor. This subsection aims to develop an efficient tracking mechanism so that each sensor can learn a better path for sending message to the robot. In the deployment phase, the robot has left a counter value as its footmark. In the patrolling phase, the robot would patrol the WSN and continuously leave a counter value at each visited sensor, as the operation executed in the deployment phase. Although the counter value can be used to trace the location of the robot by all sensors, however, the tracking path may be rugged and inefficient, as shown in Figure 1. Recall that Expression (3) asks for reducing the energy cost for packet transmission. To help the request initiators learn a better route and hence reduce the energy cost for them sending the repairing requests to the robot, an X-correction mechanism is proposed herein. After the robot completes the deployment task, it will periodically transmit the X-correction packet to adjust the counter value. Although the flooding mechanism is the simplest way to enable all sensors to maintain the up-to-date location of the robot, however, it raises significant control overheads. The proposed X-correction mechanism corrects the counter value along two cross-paths and hence significantly reduces the control overhead. The control packet broadcasted in the X-correction mechanism is called X-correction packet which aims to reduce the length of inefficient and rugged tracking path and therefore reduce the control overhead.

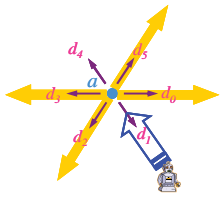

Figure 6. The basic concept of X-correction mechanism.

Let k represent the moved distance length that the robot should once again execute the X-correction mechanism. Since each sensor node has six neighbors, the robot will select four directions to broadcast the X-correction packet and hence the packet will be forwarded along X-shape paths. The selection of four directions is described in below. As shown in Figure 5, each deployed sensor has six neighboring sensors that are located in six different directions d_i, for all $0 \leq i \leq 5$. Here, we assume that the robot moves along a straight line. Therefore, when a robot intends to move to a static sensor, say a, it moves along direction d_i and then leave sensor a along direction $d_{(i+3)\text{mod}5}$. The basic concept of the X-correction mechanism is that the robot broadcasts the X-correction packet along the four directions other than the current moving and leaving directions d_i and $d_{(i+3)\text{mod}5}$. Figure 6 depicts the basic concept of the X-correction mechanism. The robot moves to and leave sensor a along directions d_1 and d_4, respectively. The broadcasted X-correction packet will be transmitted along directions d_0, d_2, d_3, and d_5. As a result, the trajectory of the X-correction packet is like an X-shape.

As shown in Figure 7, the X-correction packet consists of four fields, including the current location of the robot, Broadcast ID, Hops, and the moving direction. The Broadcast ID represents the version of the X-correction packet transmitted by the robot. The value of version is initialized by 0 and will be automatically increased by one whenever the robot broadcasts an X-correction packet.

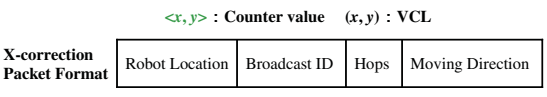

Figure 7. The format of X-correction packet.

The value of Hops is initialized by $u-1$. Upon receiving the X-correction packet, a forwarding sensor will replace its own Broadcast ID and Hops values according to the corresponding fields in the X-correction packet. Then the forwarding sensor decreases the value of Hops in the X-correction packet and subsequently rebroadcasts the revised packet.

The following presents how each sensor that receives an X-correction packet implements the X-correction mechanism. Let the robot be nearby the sensor A whose location and counter value are (m, n) and $(0, u)$, respectively. The robot periodically broadcasts an X-correction packet. Upon receiving the X-correction packet, any sensor s with location $=(x, y)$ will make a decision whether or not the packet should be forwarded. We classify the moving directions of the robot into three cases to discuss which sensors should rebroadcast the received X-correction packet. In case that sensor s should broadcast the packet, it updates its own counter value and the value of Hops field of the packet and then rebroadcasts the updated packet. The following presents the X-correction mechanism.

X-correction mechanism

```
==============================================
Let the robot broadcast an X-correction packet nearby the sensor A
whose location information and counter value are (m, n) and (0, u),
respectively.
Let node s receive X-correction packet = ((m, n), BID, Hops, d)
Case d of
    if d0, d3 then flag = true
    if d1, d4 then flag = false
    if d2, d5 then flag = true
If (flag = true)
{
    replace location information of nodes with (BID, Hops)
    update X-correction packet: Hop = Hop -1
    transmit the packet
}
```

Figure 8 gives an example to illustrate the proposed X-correction mechanism. As shown in Figure 8a, each deployed sensor maintains its original location and counter values which are set up during the network deployment phase. For example, the location and counter value of sensor a are (9, 3) and (0, 63), respectively. We assume that the robot moves close to sensor a from the direction d_1 and transmits an X-correction packet at that location. The X-correction packet contains Location $=$ (9, 3), Broadcast ID $=1$, Hops $=62$ and Directions $=d_1$. Upon receiving the packet, the neighboring sensor d does nothing since it applies the X-correction mechanism and results flag $=$ false. On the other hand, the sensor b plays a forwarder and replaces its counter value with (1, 62). Then sensor b updates the value of Hops in X-correction packet by 61 and rebroadcasts the packet. Upon receiving the packet, sensor c replaces its counter value with (1, 61). Figure 8b depicts the resultant counter values of all sensors.

4.3. Hole-healing phase: tracking the robot

In the patrolling phase, an X-correction mechanism has been applied to update the counter values of all sensors located on the X-shape lines. Upon receiving the hole-healing request, the robot switches to hole-healing phase. Two major tasks should be executed in the hole-healing phase. The first one is to efficiently receive the hole-healing request from some sensors which are nearby the holes. The other task is that the robot constructs an efficient path that passes through each hole region and heals the hole. This subsection mainly introduces how request initiators learn better routes for sending the repairing request message to the robot. The mechanism of path construction for healing the hole will be presented in the next section.

To maintain the coverage quality, whenever a sensor detects the absence of its neighboring sensor, it plays the request initiator and intends to send a request to the robot, asking for executing the redeployment task. The request initiator will initiate a repairing request packet, or RR packet in short. The RR packet contains the location of failure sensor. To deliver the RR packet to the robot along a better route, the request initiator forwards the RR packet to the neighboring sensor that has the largest counter value. Upon receiving the RR packet, the forwarder simply forwards the packet to the neighbor that has the largest counter value. We

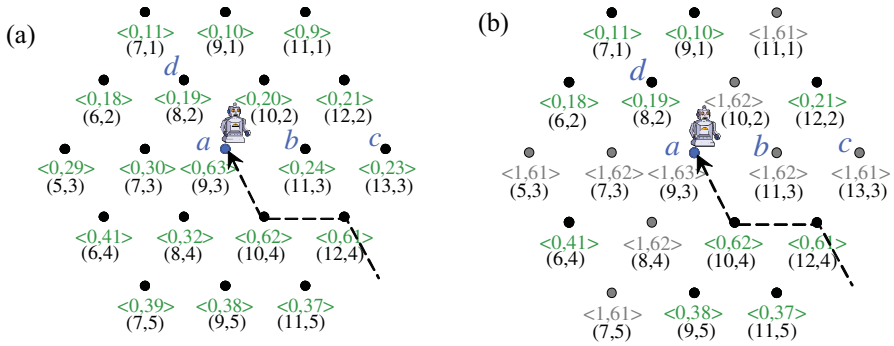

Figure 8. An example for illustrating the X-correction mechanism. (a) The original location and counter value maintained by each deployed sensor. (b) Sensors apply the X-correction mechanism.

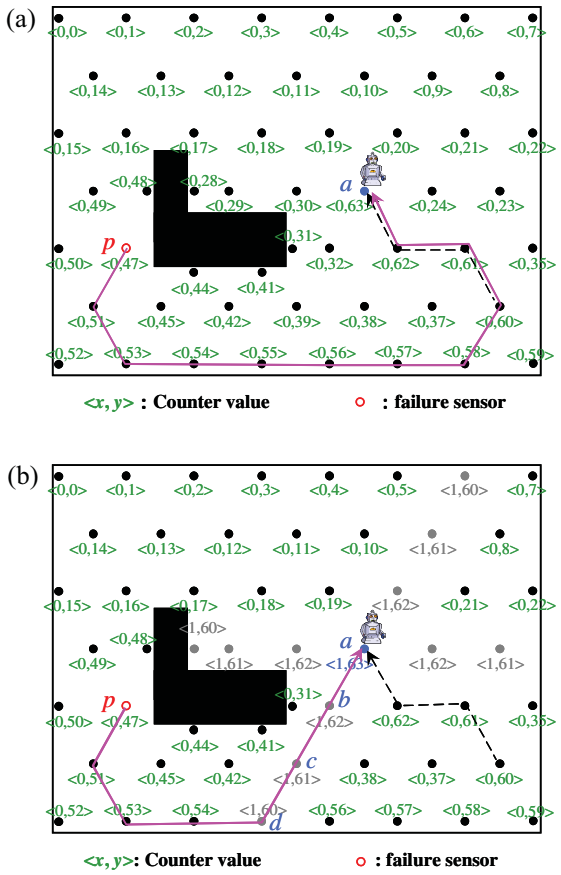

Figure 9. An example that illustrates the benefit obtained by applying the X-correction mechanism. (a) Without applying the X-correction mechanism, the RR packet is forwarded to the robot along an inefficient path. (b) Applying the X-correction mechanism to the WSN, the RR packet is delivered to the robot along an efficient route.

notice that the WSN applied the X-correction mechanism has the characteristic that the robot is always located at the sensor with the largest counter value. Consequently, the RR packet can be delivered to the robot along a better route.

Figure 9 shows an example that illustrates how a request initiator learns a better route after the X-correction mechanism has been applied on the WSN. In Figure 9a, the trajectory of the robot's movement is represented by the dotted line and the robot broadcasts an X-correction packet nearby the sensor a. We assume that sensor p detects the absence of its neighbor and plays the role of request initiator. In Figure 9a, the solid line denotes the route traversed by the RR packet in case that the X-correction mechanism is not applied to the WSN. The RR packet will be delivered to the robot by the forwarders that have the largest counter value. In fact, the packet is forwarded along the robot's trajectory. As a result, the length of route traversed by the RR packet is 11. However, by applying the proposed X-correction mechanism in Figure 9b, sensors b, c, and d have updated their counter values. Therefore, sensor d will forward the received RR packet to sensor c. Similarly, sensor c subsequently forwards the RR packet to sensor b. As a result, the length of route traversed by RR packet is 7. In comparison, applying the X-correction mechanism saves four hops for delivering the RR packet to the robot.

5. HOLE-HEALING PHASE: HOLE-HEALING MECHANISM

This section proposes a repairing algorithm for constructing an efficient path that passes through all failure regions. The proposed algorithm takes into considerations the obstacles and energy recharge issues. Herein, we notice that this paper only considers the physical obstacles existed in the monitoring region. The details of the designed repairing algorithm executed in non-obstacle and obstacle environments are presented in the following two subsections.

5.1. No obstacle environment

In the non-obstacle environment, the robot receives repairing request packets and knows the locations of failure regions and the Home. Recall that Expression (3) asks for reducing the energy cost for robot's movement. To achieve this goal, a repairing algorithm that applies dynamic programming scheme is employed herein.

Table I gives the definitions of all notations used in Figure 10 which depicts the proposed path construction algorithm. The robot will automatically perform the path construction algorithm when several requests have received or when a given time duration has been elapsed after receiving the first request. As shown in lines 1–13 of Figure 10, the maintained matrix D can help the robot construct a shortest path to visit all failure regions. After that, the robot will check whether or not it has sufficient energy to pass through all failure regions. If it is the case, the robot will directly move to repair the failure regions along the constructed path. As shown in Figure 2, the shortest repairing path that passes through failure regions A, B, C, D, and E is

Table I. The notations used in Figure 10.

N	: the number of nodes (including *Home*)
$W[]$	: adjacent matrix
R	: location of the robot
V	: set of all nodes (including *Home* and R)
U	: all subsets of V which includes *Home*
A	: set of vertices that have been processed
$D[v_i][A]$	: length of the shortest path from v_i to v_1 passing through each vertex in A exactly once
Energy (R)	: remaining energy of robot at R
E_u	: the energy consumption for the robot moving for a unit distance

Path Construction Algorithm_Without Obstacle

Void travel (**int** n, **const number** $W[]$, *minilength*, *Energy*(R)) {

```
1    index i, j, k
2    number D[1..n][subset of V-{R}]
3    for (i = 2; i <= n; i++)
4      D[i][ø]=W[i][1]
5    for (k = 1; k <= n-1; k++)
6      for (all subset A ⊆ V-{R} containing k vertices)
7      {
8        for (i such that i ≠ R and v_i is not in A)
9        {
10         D[i][A] = minimum(W[i][j]+D[j][A-{v_j}])
                     j: v_j ∈ A
11       }
12       minlength = minimum(D[j][V-R-Home])
                     v_j ∈ A, j ≠ R
13     }
14   if (Energy(R) <= minlength*E_u)
15   {
16     minlength = minimum(D[j][V-R])
17     if (Energy(R) <= minlength*E_u)
18     {
19       for (k = n-1; k >= 1; k--)
20         for (all subset A ⊆ U-{R} containing k-1 vertices)
21           if (Energy(R) >= minlength*E_u)
22           {
23             minlength = minimum(D[j][V-R])
24             break
25           }
26       }
27     }
28   }
```

Figure 10. Robot repairing algorithm that considers the remaining energy of the robot but does not consider the existence of obstacles.

constructed and is denoted by the solid line. Otherwise, in case that the remaining energy of the robot is not enough to pass through all failure regions, the home location should be visited before the robot exhausts its energy. Lines 14--27 of Figure 10 mainly cope with the energy exhaustion problem and aim to construct an efficient path that prevents the robot from energy exhaustion. As shown in line 14, if the condition $Energy(R) \leq minlength \times E_u$ holds, the remaining energy of the robot is not enough to pass through all failure regions. In this case, as shown in line 16, the home location should be visited by the robot. Lines 17–26 show that the robot figures out the suitable timing for it to visit home location. Finally, line 23 derives the shortest path, which passes through home location and visits all failure regions. As shown in Figure 3b, applying the proposed repairing algorithm, an efficient movement path $C \rightarrow A \rightarrow$ Home $\rightarrow B \rightarrow D \rightarrow E$ is constructed.

5.2. Obstacle environment

This subsection presents the path construction algorithm in considering the obstacle environment. The challenge of path construction is that the robot cannot estimate the distance between any two failure regions since there might exist obstacle between them. Moreover, an improper estimation of distance might lead to the energy exhaustion of the robot. Table II gives the definitions of all notations used in Figure 11 which depicts the proposed *Obstacle-Free Path Construction Algorithm*. First, an efficient path is constructed by applying the algorithm designed for the non-obstacle environment. However, the distance between any two failure regions might be incorrect because of the existence of obstacles. To prevent the robot from energy exhaustion, the robot should try to estimate the distance of the two successive failure regions scheduled on the constructed route. To accomplish this, as shown in line 3, the robot initiates a probe packet before its movement, aiming to estimate the distance of the route for avoiding the energy exhaustion. The probe packet will be transmitted along the constructed path and passed through all failure regions. As shown in Lines 5–9, when the probe packet arrives each request initiator, it collects information including the number of hops from the previous request initiator to the current initiator and the distance from the current initiator to the home location. Regarding the distance from the Home's location to the request initiator, it can be simply known by flooding another packet from home location over the WSN. As soon as the probe packet returns to the robot, it analyzes the collected information and determines the actual route according to the remaining energy. As shown in line 10, if the remaining energy of the robot can only reach failure region v_i, the robot goes back Home for recharging and the segment $v_i \rightarrow$ Home will be inserted in the constructed path.

Table II. The notations used in Figure 11.

obstacle_flag	:a Boolean value used to check whether or not the network contains any obstacle
simulate_energy	: remaining energy of the robot
Energy (R)	: remaining energy of robot at R

Figure 12 shows an example to illustrate the path construction algorithm applied in the obstacle environment. In Figure 12a, A, B, C, D, E, and F denote the request initiators that have sent the repairing requests to the robot and

Obstacle-Free Path Construction Algorithm

Input: repairing path (v_o, v_1, …, v_{n-1}), v_o = location of robot

```
1    if ( obstacle_flag = true)
2    {
3      transmit a probe packet to simulate the repairing path
4      simulate_energy = Energy(R)
5      for (i = 0; temp < 0; i++)
6      {
7        simulate_energy = simulate_energy – energy of v_i to v_{i+1}
8        temp = simulate_energy – energy of v_{i+1} to Home
9      }
10     repairing path = v_o, v_1, …, v_i, Home, v_{i+1}, …, v_{n-1}
11   }
```

Figure 11. Algorithm for constructing a repairing path.

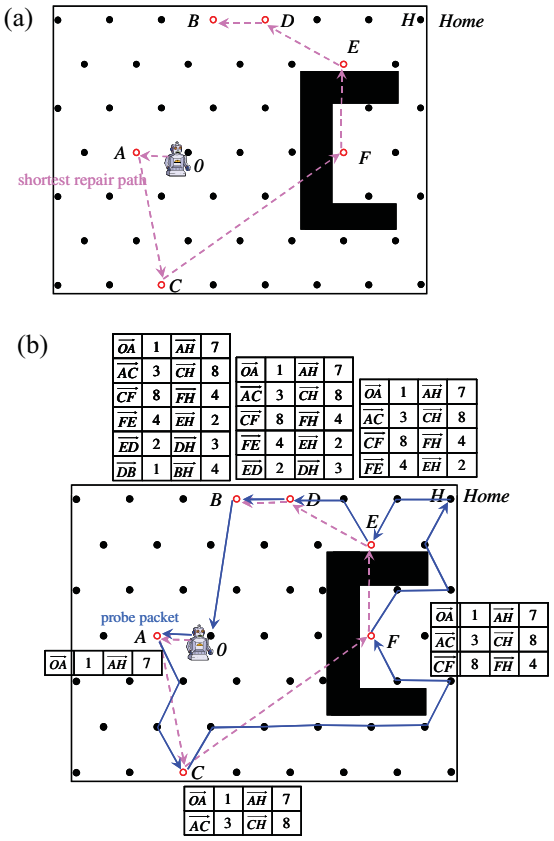

Figure 12. An example for illustrating the path construction algorithm with the consideration of obstacle. (a) The robot initiates a probe packet and transmits it along the constructed shortest path as shown in the dotted line. (b) Based on the collected information and the remaining energy, the robot constructs the repairing path which is marked by the solid line.

H denotes the Home location. Each request initiator maintains a table that records the distance between itself to the Home location. Therefore, initiators A, B, C, D, E, and F maintain $\overline{AH} = 7$, $\overline{BH} = 4$, $\overline{CH} = 8$, $\overline{DH} = 3$, $\overline{EH} = 2$, and $\overline{FH} = 4$ in their tables, respectively. Upon receiving the repairing requests from the six request initiators, the robot applies the algorithm shown in Figure 10 to construct an efficient path $A \rightarrow C \rightarrow F \rightarrow E \rightarrow D \rightarrow B$, which is represented by the dotted line as shown in Figure 12a. To estimate the length of each segment of the constructed route, the robot initiates a probe packet and transmits it along the constructed route. The probe packet contains the information of constructed path and a Hops field which counts the number of hops it traverses. Upon receiving the probe packet, the request initiator A additionally records $\overline{OA} = 1$ hop in its table according to the value of Hops field in the packet. Then request initiator A appends $\overline{OA} = 1$ and $\overline{AH} = 7$ to the probe packet. The contents of the probe packet are shown in Figure 12b. After that, initiator A transmits the probe packet along the path indicated in the probe packet. Similarly, when request initiator C receives the probe packet, it stores $\overline{AC} = 3$ hops in its table, which indicates the segment length from previous initiator A to C. The initiator C appends information including $\overline{AC} = 3$ and $\overline{CH} = 8$ to the probe packet and then transmits the packet along the path. The initiators F, E, D, and B will execute the similar operations upon receiving the probe packet. Finally, the length of each segment of the constructed route and the distance from each initiator to the home location will be known by the robot. Hence the robot is able to construct the repairing path according to its remaining energy. As shown in Figure 12b, the remaining energy of the robot can only reach to F. According to the algorithm presented in Figure 11, the robot will insert Home location between initiators F and E. As a result, the resultant movement path is $A \rightarrow C \rightarrow F \rightarrow Home \rightarrow E \rightarrow D \rightarrow B$, which is denoted by the solid line as shown in Figure 12b.

6. PERFORMANCE STUDY

This section examines the performance study of the developed tracking and repairing mechanisms. The proposed tracking and robot repairing algorithm, called TRR in short, is compared with previous work in Ref. [8] which is referred to coverage, exploration and deployment (CED) mechanism. Table III lists the parameter values which refer to the typical parameters in Berkeley motes [21].

The robot is assumed to be equipped with a compass and several 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 by 8.267 J/m which refers to previous work [22]. The experimental environment is described below. The network size is $400 \times 400\,m^2$. The home is located at the left-top corner of the monitoring region and the start location of the robot is home location. Each simulation result is obtained from the average of 100 independent runs. The 95% confidence interval is always smaller than 5% of the reported values.

The request initiator which is nearby the failure sensor will send a repairing request to the robot. Upon receiving the request message, the robot will perform the proposed repairing algorithm to construct a repairing path if it collects at least five different request messages in 10 min. In case that the robot collects fewer than five failure messages for 20 min, the robot also executes the repairing mechanism in order to prevent the request initiators from waiting for a

Table III. Simulation parameters.

Parameter	Value
Communication range	40 m
Sensing range	20 m
Packet transmission cost	0.075 J/s
Packet reception cost	0.030 J/s
Idle cost	0.025 J/s
Maximum energy consumption in motes	324 J/h
Total initial energy	32 400 J (100 h)

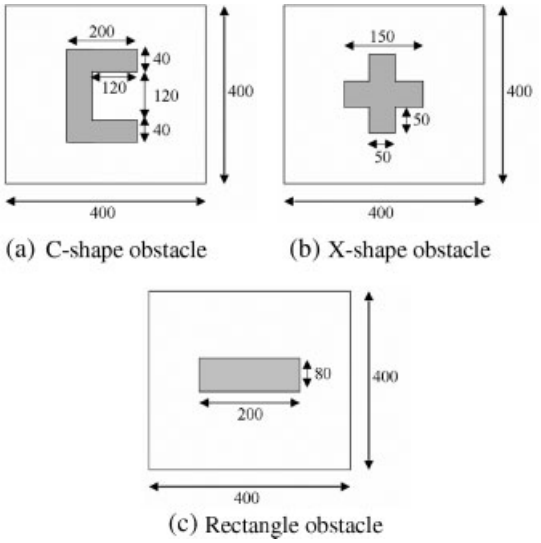

Figure 13. Obstacles with various shapes are considered in the simulation environment. (a) C-shape obstacle, (b) X-shape obstacle, and (c) rectangle obstacle.

long time. As shown in Figure 13, three different shapes of obstacles are considered in the simulation. For each obstacle shape, three different sizes of the obstacles are considered. The large size of the obstacle is shown in Figure 13. The small and middle sizes of obstacles are 40% and 70% of the large one, respectively.

The simulations evaluate the performance of repairing mechanisms in terms of the recovery delay time, coverage ratio, and the number of sensor nodes deployed for repairing WSN by the robot for each movement. An X-correction mechanism is proposed to help the request initiator learn a better route. To evaluate the benefit obtained from the proposed X-correction, two straightforward mechanisms including flooding-correction mechanism and line-correction mechanism are compared with the proposed X-correction mechanism in terms of the control packet overhead and the length of tracking path. In the flooding mechanism, the robot floods a correction packet over the entire WSN and all sensor nodes update their counter values according to the correction packet. In the line-correction mechanism, the robot broadcasts a correction packet to those nodes that have the same y-coordinate value with the robot.

We assume that the robot broadcasts the correction packet for every movement of 400 m. The failure nodes are randomly selected and five failure nodes will be generated every 10 min. As shown in Figure 14, the line-correction and flooding mechanisms have smallest and largest control overheads, respectively. The control packet overhead of X-correction mechanism is approximately 2.5 times of the line-correction mechanism while the control overhead of flooding mechanism is about 37 times of X-correction mechanism. In other words, the control overheads of the line-correction and X-correction are similar and they are significantly smaller than that of the flooding mechanism.

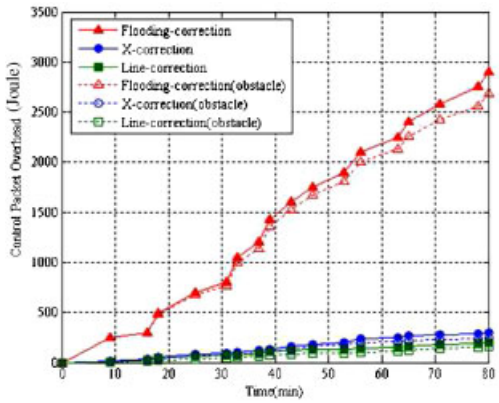

Figure 14. The control overhead of three mechanisms for correcting the counter values.

To evaluate the impact of applying the three correction mechanisms on the tracking length, the average length of tracking path is calculated and compared. Figure 15 examines the average distance between sensor nodes and robot. The repairing request packet sent by any sensor node can always reach the robot by traveling the shortest path by applying the flooding-correction mechanism. Although the lengths of tracking path by applying the X-correction and line-correction mechanisms are not as short as the one created by applying the flooding mechanism, however, the X-correction mechanism efficiently reduces the average length and keeps the path length smaller than 520 m. As a result, the average path lengths by applying X-correction and line-correction mechanisms are 370 and 483 m, respectively.

The following introduces the *Correction Efficient Index* (*CEI*) to estimate the contribution of each control overhead in terms of reduced path length. Let $|L_{none}|$ denote the length of tracking path L_{none} without applying any correction mechanism. That is, the initiator sends the repairing packet to the robot along the path that the robot traversed. As a result, the path L_{none} is rugged and inefficient, as shown in

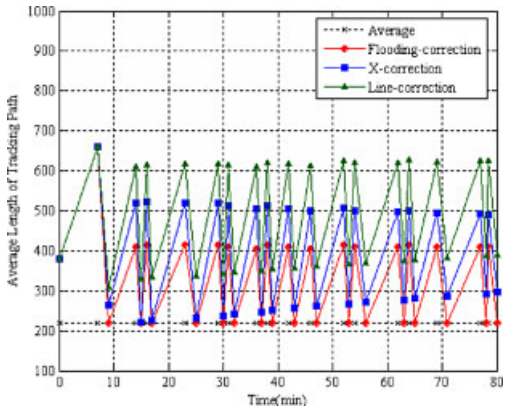

Figure 15. The average length of tracking path from request initiators to the robot by applying the three correction mechanisms.

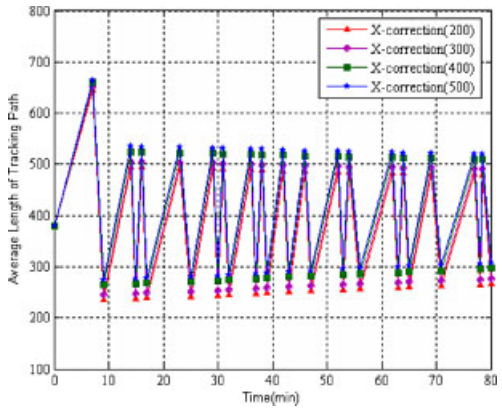

Figure 16. The average length of tracking path by varying the threshold value of movement distance in the X-correction mechanism.

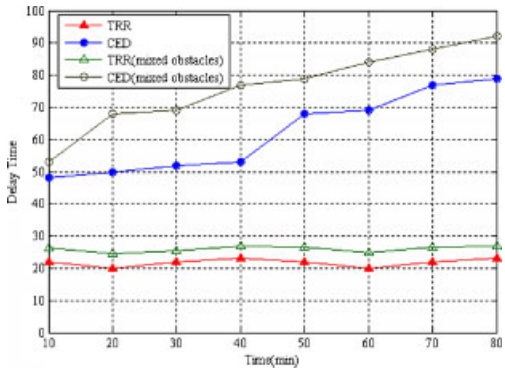

Figure 17. Comparison of TRR and CED in terms of recovery delay in the environment that contains multiple obstacles with different shapes.

the blue path of Figure 1. Let L_i^* and J_i denote the tracking path and control packet overhead, respectively, by applying the mechanism i, where $i \in \{f$: flooding-correction, x: X-correction, l: line-correction$\}$. The CEI_i is formulated as Equation (4):

$$CEI_i = \frac{|L_{none}| - |L_i^*|}{J_i} \tag{4}$$

The mechanism that results in a largest value of CEI would be the most cost-effective mechanism. According to Figure 14, the average control packet overheads by applying the flooding-correction, X-correction, and line-correction mechanisms are $J_f = 2874$ J, $J_x = 316$ J, and $J_l = 287$ J, respectively. As shown in Figure 15, the average path lengths by applying the flooding-correction, X-correction, and line-correction mechanisms are $L_f^* = 314$ m, $L_x^* = 370$ m, and $L_l^* = 483$ m, respectively. Since the value of $|L_{none}|$ is 1033 m in our simulation, we can derive the values of CEI_f, CEI_x, and CEI_l are 0.25, 2.1, and 1.92, respectively, by applying Equation (4). As a result, the X-correction mechanism has the largest CEI value and its control overhead is the most cost effective.

The threshold of robot movement distance that initiates the X-correction mechanism is ranging from 200 to 500 m. As shown in Figure 16, the value of threshold set by 200 and 300 m can reduce the average length of tracking path. When the threshold values are 400 and 500 m, the improvement of path length is not significant. Therefore, the X-correction mechanism will use 400 m as the threshold value to compare with the CED mechanism [8] in the repairing task.

Figure 17 compares TRR and CED in terms of recovery delay. In the environment, multiple middle-sized obstacles with different shapes are randomly generated. The recovery delay time for repairing failure regions required in TRR is smaller than that of CED even through there exist multiple obstacles. The major reason is that TRR actively notifies the robot to repair the failure regions while CED passively waits for repairing until the robot visits that region. The CED did not concern the delay-constraint issue for repairing the failure sensors. The failure sensors can be repaired by the robot only when the robot moves to their locations. Therefore, CED did not guarantee the recovery delay. In addition, the repairing algorithm proposed in TRR constructs a shortest path for robot's movement in the environment without obstacles and hence reduces the time required for repairing the failure regions. When the environment contains multiple obstacles, the robot's movement will be blocked and the repair path may be rugged. The proposed repairing algorithm takes the obstacles into consideration and constructs an efficient path which significantly reduces the required repairing time. In comparison, TRR outperforms CED in terms of recovery time.

Figure 18 investigates the impacts of obstacle size on recovery delay time. The TRR maintains a constant delay time in all cases because the probe packet helps the robot to construct an efficient repair path when the environment contains obstacles. The recover time of the CED will increases with the obstacle size. The reason is that the robot's movement is easier to be blocked by the large-size obstacle and hence the CED increases the required time for repairing the

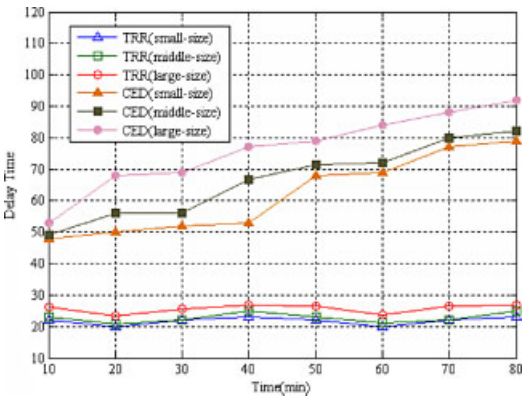

Figure 18. Comparison of TRR and CED in terms of recovery delay in the environment that contains multiple obstacles with different sizes.

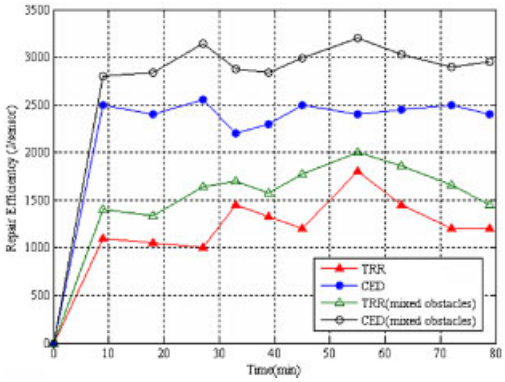

Figure 19. Comparison of TRR and CED in terms of the repair efficiency in the environment that contains multiple obstacles with different shapes.

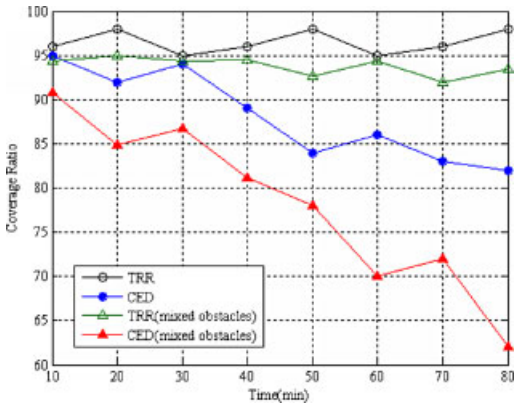

Figure 21. Comparison of TRR and CED in terms of the coverage ratio in the environment that contains multiple obstacles with different shapes.

failure regions. As a result, TRR reduces the recovery delay time and outperforms the CED in all cases.

In addition to the network recovery time, the proposed TRR additionally reduces the energy consumption of the robot. When the robot receives multiple repairing request messages, it constructs a shortest route that passes through all failure regions. Even though the environment contains multiple obstacles with different shapes, TRR also constructs an efficient route along which the robot saves its energy for movement. Figure 19 measures the repairing efficiency which is defined by the average energy consumption required for robot to repair a failure region. Since the robot moves along the better route, the TRR has a better repairing efficiency than CED.

The existence of obstacles increases the difficulty of path construction because the robot cannot estimate the distance between two failure regions. The possibility that the obstacle exists between two failure regions increases with the obstacle size. As shown in Figure 20, TRR outperforms CED in terms of the average energy consumption required for robot to repair a failure region. By applying the probe packets, TRR estimates the distance of the route and prevents the robot from energy exhaustion. The TRR helps the robot construct an efficient repairing path and therefore maintains 1500 of repair efficiency in average. On the other hand, CED passively waits for repairing until the robot visits the region. Consequently, TRR has a better repair efficiency than CED.

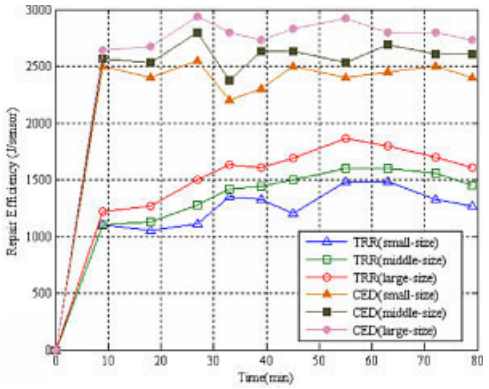

Figure 20. Comparison of TRR and CED in terms of the repairing efficiency in the environment that contains multiple obstacles with different sizes.

Moreover, we compare the TRR and CED in terms of the coverage ratio in the environment that contains multiple obstacles with different shapes. The proposed TRR enables request initiators to actively notify the robot to repair the failure region, resulting in the failure region being rapidly redeployed within 10 min. Therefore, TRR maintains above 90% of coverage ratio, even through there are multiple obstacles with different shapes. However, CED passively waits the robot to repair the failure region, and hence some existing holes cannot be repaired in time. In the worst case, CED maintains 62% of coverage ratio. As shown in Figure 21, TRR has a better coverage ratio than CED.

Figure 22 depicts that the coverage ratio of TRR is larger than that of CED in the environment containing multiple obstacles with different sizes. The TRR maintains above 95% of coverage ratio in all cases since the robot moves along an efficient path and hence the failure regions can be repaired in time. Finally, the TRR maintains a similar coverage ratio. On the other hand, the CED locally guides the robot by the nearby sensors and thus the resultant repair path is rugged and inefficient. The number of sensor failures increase with mission time. This also indicates that the number of coverage holes increases with the working time. In general, the lengths of repair paths by applying TRR and CED increase with the size of the obstacle. Therefore, both TRR and CED have smaller coverage ratio in the environment that contains a lager size of obstacle.

Finally, we compare the TRR and CED in terms of variation of repair efficiency by changing the home locations. Note that a large value of variation of repair efficiency represents that the path length constructed by the applied

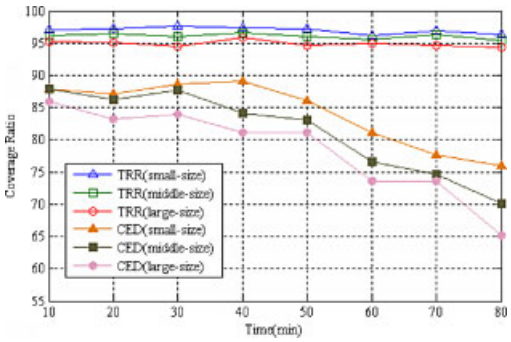

Figure 22. Comparison of TRR and CED in terms of the coverage ratio in the environment of containing multiple obstacles with different sizes.

algorithm highly depends on the home location. Herein, as shown in Figure 23a, we equally partition the monitoring region into nine subregions. The central points of the nine subregions, denoted by UL, U, UR, L, C, R, DL, D, and DR, are the candidates of home location. The CED does not schedule the repair path for the failure regions and thus the robot might need to charge its energy after repairing one failure region which is far away the home location, resulting in an inefficient path. Therefore, the path length of CED highly depends on the home location. In Figure 23b, the variation of the repair efficiencies of CED and TRR are 700 and 300, respectively. As a result, the TRR has a smaller variation of the repair efficiency than CED. The major reason is that the proposed repairing algorithm takes into consideration the home location and therefore the path length constructed by TRR will not be affected significantly by the home location. As a result, the TRR outperforms CED in terms of the variation of repair efficiency when the home location is varied.

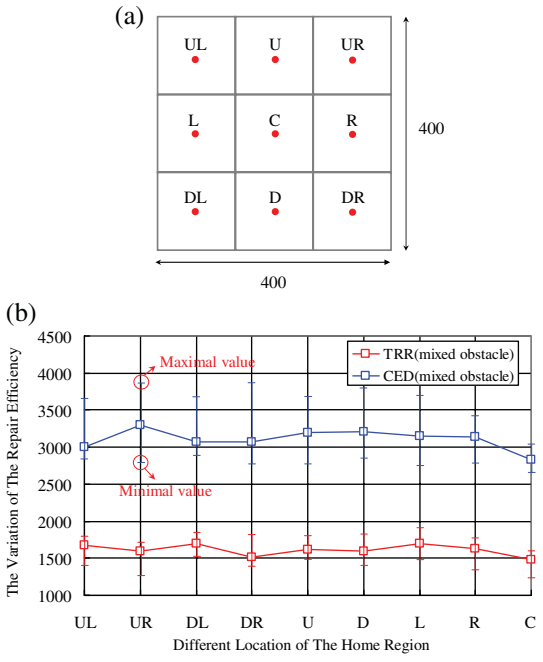

Figure 23. Comparison of TRR and CED in terms of the variation of repair efficiency when the home location is changed.

7. CONCLUSIONS

This paper proposes a tracking mechanism and a robot repairing mechanism for maintaining coverage quality of a given WSN. With the extension of the existing robot deployment algorithm [7–11], the robot leaves location information and counter value on each deployed sensor as a footmark for tracking the robot. After the robot completes its deployment task, it moves for handling the event and repairing the failure regions. Since the robot changes its location, an X-correction mechanism is proposed for the robot to update the counter values of some sensors in a cost-effective manner. Consequently, the request initiator learns a better route to notify the robot for repairing. Upon receiving the location of failure regions from multiple request initiators, the robot dynamically constructs a shortest route that passes through each failure region for executing the redeployment task so that the coverage quality of the given WSN can be maintained. The route construction also takes obstacle and the constraint of robot's remaining energy into consideration. Performance results reveal that the proposed tracking and repairing algorithms outperform existing mechanism in terms of recovery delay time and coverage ratio.

REFERENCES

1. Krysander M, Frisk E. Sensor placement for fault diagnosis. *IEEE Transactions on Systems, Man and Cybernetics, Part A* 2008; **38**(6): 1398–1410.
2. Chen A, Lai TH, Xuan D. Measuring and Guaranteeing Quality of Barrier-Coverage in Wireless Sensor Networks. *ACM International Symposium on Mobile Ad Hoc Networking and Computing* (*MobiHoc*), Hong Kong, June 2008.
3. Kay JM, Frolik J. An expedient wireless sensor automaton with system scalability and efficiency benefits. *IEEE Transactions on Systems, Man and Cybernetics, Part A* 2008; **38**(6): 1198–1209.
4. Kleinberg R. Geographic Routing Using Hyperbolic Space. *IEEE International Conference on Computer Communications* (*INFOCOM*), Alaska, USA, May 2007.
5. Zhang W. A probabilistic approach to tracking moving targets with distributed sensors. *IEEE Transactions on Systems, Man and Cybernetics, Part A* 2007; **37**(5): 721–731.

6. Glasgow JM, Thomas G, Pudenz E, Cabrol N, Wettergreen D, Coppin P. Optimizing information value: improving rover sensor data collection. *IEEE Transactions on Systems, Man and Cybernetics, Part A* 2008; **38**(3): 593–604.
7. Batalin MA, Sukhatme GS. Efficient Exploration without Localization. *IEEE International Conference on Robotics and Automation* (*ICRA*), Taipei, Tanwan, May 2003.
8. Batalin MA, Sukhatme GS. Coverage, Exploration and Deployment by a Mobile Robot and Communication Network. *International Workshop on Information Processing in Sensor Networks*, 376–391, Palo Alto Research Center (*PARC*), Palo Alto, April 2003.
9. Batalin MA, Sukhatme GS, Hattig M. Mobile Robot Navigation Using a Sensor Network. *IEEE International Conference on Robotics & Automation* (*ICRA*), New Orleans, LA, April 2004.
10. Chang CY, Chang HR, Hsieh CC, Chang CT. OFRD: Obstacle-Free Robot Deployment Algorithms for Wireless Sensor Networks. *IEEE Wireless Communications and Networking Conference* (*WCNC*), Hongkong, March 2007.
11. Batalin MA, Sukhatme GS. The design and analysis of an efficient local algorithm for coverage and exploration based on sensor network deployment. *IEEE Transactions on Robotics* 2007; **23**(4): 661–675.
12. Mataric MJ. Behavior-based control: examples from navigation, learning, and group behavior. *Journal of Experimental and Theoretical Artificial Intelligence, special issue on Software Architectures for Physical Agents* 1997; **9**(2–3): 323–336.
13. Pirjanian P. Behavior Coordination Mechanisms-State-of-The- Art. *Tech. Rep. IRIS-99-375, Institute for Robotics and Intelligent Systems, University of Southern California*, October 1999.
14. Zou Y, Chakrabarty K. Sensor deployment and target localization in distributed sensor networks. *ACM Transations on Embedded Computing Systems* 2004; **3**: 61–91.
15. Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shamoys DB (eds). The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, John Wiley & Sons: New York.
16. Balas E. New classes of efficiently solvable generalized traveling salesman problems. *Annals of Operations Research* 1999; **86**: 529–558.
17. Yan XS, Liu HM, Yan J, Wu QH. A Fast Evolutionary Algorithm for Traveling Salesman Problem. *International Conference on Natural Computation* (*ICNC*), Hainan, China, August 2007.
18. Shang Y, Ruml W, Zhang Y. Improved MDS-Based Localization. *IEEE International Conference on Computer Communications* (*INFOCOM*), Hongkong, March 2004.
19. Chang CY, Chang CT, Chang SW, Chen YC, Lee MH. Path Guiding Mechanisms for Mobile Anchor Improving or Balancing Location Accuracies of Static Sensors in WSNs. *IEEE Conference on Local Computer Networks* (*LCN*), Montreal, Canada, September 2008.
20. Xiao B, Chen H, Zhou S. A Walking Beacon-Assisted Localization in Wireless Sensor Networks. *IEEE International Conference on Communications* (*ICC*), Glasgow, Scotland, June 2007.
21. Hill J, Culler D. A Wireless Embedded Sensor Architecture for System-Level Optimization. *Technical report, Computer Science Department, University of California at Berkeley*, 2002.
22. Ganeriwal S, Kansal A, Srivastava MB. Self Aware Actuation for Fault Repair in Sensor Neworks. *IEEE International Conference on Robotics and Automation* (*ICRA*), Barcelona, Spain, April 2004.

AUTHORS' BIOGRAPHIES

Chih-Yung Chang received the Ph.D. degree in Computer Science and Information Engineering from National Central University, Taiwan, in 1995. He joined the faculty of the Department of Computer and Information Science at Aletheia University, Taiwan, as an Assistant Professor in 1997. He was the Chair of the Department of Computer and Information Science, Aletheia University, from August 2000 to July 2002. Since August 2002, he has been an Associate Professor with the Department of CSIE, Tamkang University. He is currently a Full Professor with the Department of CSIE at Tamkang University, Taiwan. Dr. Chang served as an Associate Guest Editor of *IET Communications* (2011), *Telecommunication Systems* (TS, 2010), *Journal of Information Science and Engineering*(JISE, 2008), *Journal of Internet Technology*(JIT, 2004 & 2008), *Journal of Mobile Multimedia*(JMM, 2005), and a member of Editorial Board of *Tamsui Oxford Journal of Mathematical Sciences* (2001–2008) and *Journal of Information Technology and Applications* (JITA, 2008). He was an Area Chair of *IEEE AINA*'2005, Vice Chair of *IEEE WisCom*'2005 and *EUC*'2005, Track Chair (Learning Technology in Education Track) of *IEEE ITRE*'2005, Program Co-Chair of *WASN*'2007, *MNSAT*'2005 and *UbiLearn*'2006, Workshop Co-Chair of *ICS*'2008, *INA*'2005, *MSEAT*'2003, *MSEAT*'2004, and Publication Chair of *MSEAT*'2005 and *SCORM*'2006. Dr. Chang is a member of the IEEE Computer Society, Communication Society and IEICE society. His current research interests include wireless sensor networks, Bluetooth radio networks, Ad Hoc wireless networks, and WiMAX broadband technologies.

Chih-Yu Lin received the B.S. degree in Computer Science and Information Engineering from Aletheia University, Taiwan, in 2007, and the M.S. degree in Computer Science and Information Engineering from Tamkang University, Taiwan, in 2009. Since 2009, he was working toward his Ph.D. degree in Department of Computer Science and Information Engineering at Tamkang University. Mr. Lin won lots of scholarships in Taiwan and participated in many Wireless Sensor Networking projects. His research domains include wireless sensor networks, Ad-Hoc wireless networks, mobile/wireless computing, and WiMAX.

Gwo-Jong Yu received the BS degree in Computer Science from the Chung-Yuan Christrian University, Taiwan in 1989, and the PhD degree in Computer Science from the National Central University, Taiwan in 2001. Since August 2001, he is an assistant professor in the department of computer and information science, Aletheia University, Taiwan. He is a member of IEEE. Since August 2006, he became an associate professor in the department of computer science and information engineering, Aletheia University, Taiwan. His current research interests include wireless sensor networks, ad hoc networks, WiMAX and LTE.

Chin-Hwa Kuo received his Ph.D. in electrical engineering from University of Notre Dame, South Bend, IN. USA. in 1994. Currently, he is a professor and chairman of Department of Computer Science and Information Engineering and director of Center for Digital Language Research at Tamkang University, Taiwan ROC. His research interests include multimedia processing and applications, technology enhanced language learning, social computing in education.

DOI: 10.1002/wcm