Enhanced Deployment Algorithms for Heterogeneous Directional Mobile Sensors in a Bounded Monitoring Area

Ting-Yu Lin, *Member, IEEE*, Hendro Agus Santoso,
Kun-Ru Wu, *Student Member, IEEE*, and Gui-Liu Wang

Abstract—Good deployment of sensors empowers the network with effective monitoring ability. Different from omnidirectional sensors, the coverage region of a directional sensor is determined by not only the sensing radius (distance), but also its sensing orientation and spread angle. Heterogeneous sensing distances and spread angles are likely to exist among directional sensors, to which we refer as heterogeneous directional sensors. In this paper, we target on a bounded monitoring area and deal with heterogeneous directional sensors equipped with locomotion and rotation facilities to enable the sensors self-deployment. Two Enhanced Deployment Algorithms, EDA-I and EDA-II, are proposed to achieve high sensing coverage ratio in the monitored field. EDA-I leverages the concept of virtual forces (for sensors movements) and virtual boundary torques (for sensors rotations), whereas EDA-II combines Voronoi diagram directed movements and boundary torques guided rotations. EDA-I computations can be centralized or distributed that differ in required energy and execution time, whereas EDA-II only allows centralized calculations. Our EDA-II outperforms EDA-I in centralized operations, while EDA-I can be adapted into a distributed deployment algorithm without requiring global information and still achieves comparably good coverage performance to its centralized version. To the best of our knowledge, this is perhaps the first work to employ movements followed by rotations for sensors self-deployment. Performance results demonstrate that our enhanced deployment mechanisms are capable of providing desirable surveillance level, while consuming moderate moving and rotating energy under reasonable execution time.

Index Terms—Directional sensors deployment, sensing coverage, virtual boundary torques, wireless sensor network

1 Introduction

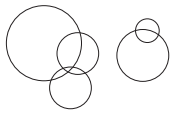
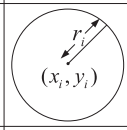
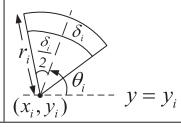
A wireless sensor network (WSN) is widely used for habitat and environmental surveillance, medical application (with the purpose of improving quality of health care), agricultural assistance, and as solutions to military problems [10], [16], [23], [24]. Wireless sensors generally come in two sensing shapes: omnidirectional (disk-shaped) and directional (fan-shaped). As illustrated in Table 1, omnidirectional sensors obey the circular sensing model, while directional sensors have sector-like sensing behavior. An evident difference between the two types of sensors is that a directional sensor is identified by both its location (position) and sensing direction (orientation). Another difference in terms of sensing coverage is that, *the covered region of a directional sensor is determined by not only the sensing radius, but also its sensing spread angle*. Consequently, an arbitrary directional sensor s_i is generally represented by a 5-tuple $(x_i, y_i, r_i, \theta_i, \delta_i)$, where (x_i, y_i) denotes the coordinate (location) of s_i, r_i expresses the sensing radius, θ_i points out the viewing angle (sensing orientation), and δ_i indicates the sensing spread angle. In this paper, the viewing angle θ_i is defined as the angle this sensor is facing with respect to the horizontal line of $y = y_i$ in counterclockwise direction, where $0 \leq \theta_i < 2\pi$.

For a successful network surveillance (whether it is omnidirectional or directional WSN), providing sufficient sensing coverage is essential. Manual placement of static sensors involves manpower effort and lacks network self-healing competence (when faulty sensors occur). Thanks to the availability of motion and rotation facilities, we consider smart sensors with movable and rotatable capabilities to accomplish self-deployment after an initial random placement of sensors. Envision a public indoor place where valuable items/collections are kept, such as an exhibition hall (holding precious pieces of artwork) inside a museum, as illustrated in Fig. 1. During the daytime (opening hours), traditional closed-circuit (static) television cameras may be sufficient for monitoring the area. However, during the nighttime when visitors are no longer allowed inside and only limited on-duty guards are available, a self-deployable mobile visual sensor network would be useful for maintaining full surveillance of those exhibition halls. Two main benefits come with the employment of a self-deployable mobile visual sensor network. First, the mobile visual sensors (cameras) are capable of self-deploying themselves without the effort of manual placements. This deployment flexibility facilitates the initial network configuration and still ensures a high confidence in terms of achieved

• *The authors are with the Department of Electrical and Computer Engineering National Chiao Tung University, Hsinchu, Hsinchu, Taiwan. E-mail: ting@cm.nctu.edu.tw, {ndro2.eed01g , wufish.cm96g}@nctu.edu.tw, willian960@gmail.com.*

Manuscript received 29 June 2014; revised 30 Nov. 2015; accepted 2 May 2016. Date of publication 5 May 2016; date of current version 2 Feb. 2017. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TMC.2016.2563435

TABLE 1
Classification of Sensors Based on Circular (Omnidirectional) or Sector-Like (Directional) Sensing Shapes

Attribute / Type	Sensing Shape	Sensing Capability	Sensor Representation
Type-I (disk-shaped)		temperature, humidity, brightness, etc.	 (x_i, y_i): location of sensor s_i r_i : sensing radius of s_i
Type-II (fan-shaped)		camera, infrared (radar), ultrasound (sonar), etc.	 (x_i, y_i): location of sensor s_i r_i : sensing radius of s_i θ_i : viewing angle of s_i δ_i : spread angle of s_i

surveillance (coverage) level (ratio). Second, by developing good self-deployment algorithms, mobile visual sensors (cameras) can perform re-deployment dynamically in the face of faulty sensors occurring. This automatic re-deployment ability empowers the visual sensing system with greater surveillance capacity for detecting intrusions (from outside) or emergency events (from inside) than its traditional static sensing counterpart. Fig. 1a displays an exhibition hall at night when security guards are absent to patrol the area with mobile visual sensors initially scattered randomly across the region. After performing some self-deployment algorithm, mobile visual sensors reposition themselves and the surveillance coverage has been effectively enhanced/improved, as shown in Fig. 1b. Similar scenarios/applications can also be found in other indoor bounded environments, such as factory spaces (for keeping expensive equipment), shopping malls, business offices (for storing confidential documents), school labs, etc., for safety and security concerns. Our goal is to develop good sensors self-deployment algorithms to effectively enhance the sensing coverage of the target environment.

The remainder of this paper is organized as follows. Section 2 reviews several previous research efforts and summarizes our unique contributions. In Section 3, we present the environmental assumptions made in this work. Our EDA-I and EDA-II deployment algorithms are elaborated in Sections 4 and 5, respectively. EDA-I computations can be centralized or distributed that differ in required energy and execution time, whereas EDA-II only allows centralized calculations. Section 6 presents the performance results, with respect to obtained sensing coverage and total energy consumed by physical sensors movements and rotations. Our EDA-II outperforms EDA-I in centralized operations, while EDA-I can be adapted into a distributed deployment algorithm without requiring global information and still achieves comparably good coverage performance to its centralized version. Finally, we draw our conclusion in Section 7.

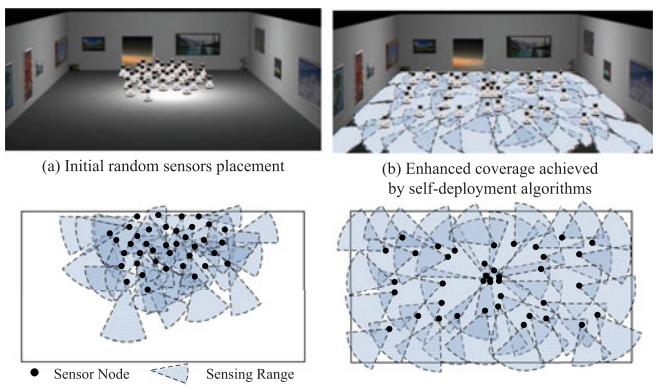


Fig. 1. Illustration of an exhibition hall inside a museum at night, where (a) shows the initial randomly-placed sensors and (b) displays the enhanced surveillance coverage achieved by self-deployment algorithms.

2 Related Work

A number of research efforts have explored the movement-assisted sensors deployment techniques by utilizing mobile sensors to enhance the sensing coverage [26], [31], [32], [38]. However, those works target on omnidirectional sensors.

Due to the increasing popularity of surveillance camera networks and video sensing applications[1], several research works on the coverage problem of directional sensor networks have recently emerged [8], [11], [21], [22], [28], [36], [37]. Among those, [21] has been perceived as the first work to study coverage problems in directional (video) sensor networks. In [11], the authors introduce a distributed algorithm that schedules the sensing directions (orientations) of active sensors to maximize the covered area. Also studying the coverage problem for directional sensors with tunable orientations, [8] proposes scheduling algorithms to minimize the set of active sensors while maximizing the number of monitored targets. Taking another perspective on the coverage problem, authors in [22] study how many sensors N are needed to meet a given required coverage probability p, in a directional sensor network where sensors are uniformly distributed. In addition, given fixed sensor population N, the problem of how sensing radius r should be adjusted is also discussed. However, the above works do not tackle the sensors deployment problem directly. To address the deployment problem, authors in [28], [36], [37] propose coverage-aware algorithms for deploying directional sensors by rotating sensors toward coverage holes. A more detailed survey on the coverage issues for directional sensors can be found in [13]. The aforementioned research works deal with homogeneous directional sensors in their approaches. Nevertheless, heterogeneity in sensing radius and spread (offset)

1. In this paper, we use standard directional camera sensors to provide quality surveillance images/pictures for our envisioned applications.

angle commonly exists among directional sensors.[2] Thus deployment algorithms that consider heterogeneous directional sensors are practically necessary, despite the design challenges. Several research efforts in the area of multimedia sensor networks (MSNs) work with heterogeneous sensors [29], [30], [34], [35]. The authors in [34] investigate how many heterogeneous multimedia sensors are necessary to ensure a certain type of monitoring service under a hybrid of omnidirectional and directional sensing models, while [35] studies the omnidirectional audio sensors deployment problem to achieve *k*-class coverage for sensing reliability. The above research directions differ from our design goal targeted in this paper for achieving high coverage ratio under a given number of heterogeneous visual directional sensors. On the other hand, [29], [30] discuss the barrier coverage problem (detection/sensing ability crossing a barrier) for visual directional sensors. The barrier coverage issue is also different from the area coverage problem studied in this paper.

By observing rotations alone cannot sufficiently improve the coverage performance, two recent works have appeared to leverage both sensors rotations and movements in their designs [14], [27]. In [14], the authors introduce the term "motility" as the rotational capability (adjustment of sensing angles) of directional sensors, and propose a motility-assisted mobility (MAM) deployment algorithm to enhance the area coverage. The MAM approach exercises sensors movements *after* sensors rotations in a cascaded manner. Based on the info of uncovered region obtained from local neighborhood, an attractive force map is constructed to determine the working directions (sensing angles) of sensors. Sensors movements are then performed (*after rotations*) through a window-scanning process, which often nullifies the previously calculated working directions and consumes significant amount of computation time. In addition, the MAM mechanism causes many sensors to travel long distances, thus consuming significant energy resource as well. In [27], the authors introduce a distributed Voronoi-based self-redeployment algorithm (DVSA), which guides sensors to new locations and new sensing directions without global information by utilizing the features of constructed Voronoi polygons/cells. The proposed DVSA determines the target position to be the vertex of the corresponding Voronoi cell with the largest angle and the sensing direction to maximize the intracell coverage. With only local geometrical info available, the coverage performance of DVSA is limited, especially when sensors are unevenly distributed with several overlapped/concentrated sensing regions existing in the monitoring environment. Although we are not the first to propose deployment algorithms that mesh sensor rotations with movements, to the best of our knowledge, we are perhaps the first to employ movements followed by rotations for sensors self-deployment. In other words, unlike [14] that exercises rotations before movements, we propose that *rotations should be applied* each time *after proper movements* have been performed to further improve the sensing coverage.

In this paper, we do *not* intend to address the issue of required number of sensors for achieving certain degree of sensing coverage. Rather, given any number of sensors, we investigate the *heterogeneous directional sensors deployment problem* by proposing algorithms to improve the coverage ratio after a possibly random deployment of sensors. Tapping into the movable and rotatable capabilities enabled by locomotion and revolving facilities geared on sensors, we develop self-deployment algorithms with the goal of achieving high (desirable) coverage ratio in a bounded monitoring area.

2.1 Our Contributions

Several unique contributions are made in this work. First, we observe that *rotating directional sensors at fixed positions is not as effective as re-distributing them to adequate positions, in terms of extending sensing coverage*. Instead of merely revolving sensors as in [28], [36], [37], our deployment approaches translocate sensors with reasonable traveling distances.[3] Rotations are applied each time after proper movements have been performed to further improve the sensing coverage. To the best of our knowledge, this is perhaps the first work to employ movements then rotations for sensors self-deployment. Second, since heterogeneity of directional sensors includes both the different sensing distances and varying spread angles, simultaneously taking both parameters into account complicates the protocol design. Therefore we propose to create *virtual omnidirectional sensors* with circular sensing shape that approximates the original sensing sector. Different combinations of sensing distance and spread angle in directional sensors now map to distinct values of sensing radius in the created virtual omnidirectional sensors. In this way, *we successfully transform the two-parameter deployment function into a single-parameter method*. Then our deployment algorithms operate on those virtual sensors and guide the actual directional sensors to self-deploy themselves accordingly. Third, *a novel concept of virtual boundary torques is introduced to assist in sensors rotations*. The basic idea of exercising virtual boundary torques (VBT) is to keep sensors staying inside the monitoring region and facing outward. Our deployment algorithms leverage this concept and further enhance the coverage performance, while pleasantly conserves total energy consumption by preventing sensors from traveling far.

3 Environmental Assumptions

In this paper, we focus on the deployment problem in a bounded monitoring area, which can be generally approximated as a rectangular region. The boundaries do *not* refer to

2. To deploy visual directional sensors in the monitored environment, commodity (off-the-shelf) camera sensors are generally utilized for surveillance purposes. Current camera technologies impose a certain range of working/viewing directions (sensing angles) based on the type of focal lens used. Standard FOV (spread angle) typically ranges from 45 to 60 degrees (as documented in [1, 2]). On the other hand, sensing/viewing radius/distance is related to the type of used focal lens and required/desired image resolution, so-called pixel-per-foot (PPF). Detailed calculations of the standard viewing distances can be found in [7]. Consequently, by conveniently mixing commodity camera sensors in the monitored environment, sensors heterogeneity in sensing/detection distance and spread angle (FOV) can be commonly observed among standard camera sensors.

3. Here "reasonable" aims to reflect a nice feature that our deployment algorithms are capable of translocating sensors to effectively improve the coverage ratio while preventing sensors from traveling far away, thus consuming "moderate" energy resource compared to other implemented deployment approaches. Detailed performance results to demonstrate this feature can be found later in Section 6.

physical walls that prohibit sensors from moving outside the monitored zone. Rather, the boundaries define *an area of interest*, and sensors do not necessarily always stay inside the monitored zone after performing the self-deployment algorithms. However, the sensing coverage outside the monitored zone is wasted since we are not interested. Therefore in the proposed algorithm, we aim to reduce such wasted coverage and increase the effective *sensing coverage ratio*, which is defined as the percentage of area covered by at least one sensor in the bounded monitoring region. Below we summarize our environmental assumptions.

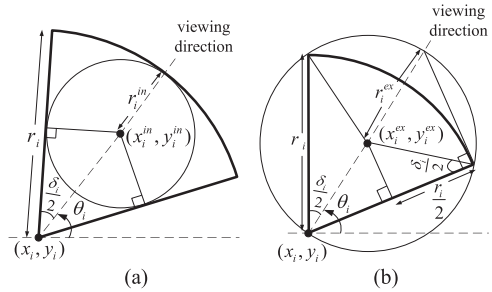


Fig. 2. Obtaining (a) the inscribed circle and (b) the circumscribed circle of a fan-shaped sensor.

(A1) There exists a powerful clusterhead responsible for performing centralized computations. All sensors are able to communicate with the clusterhead via single-hop or multi-hop wireless transmissions.

(A2) Sensors have sector-like sensing shapes and the binary sensing/detection behavior[4], in which an event is detected (not detected) by a sensor with complete certainty if this event occurs inside (outside) its sensing coverage. Both the homogeneous and heterogeneous sensors are allowed in our model. Information of respective sensing distances and spread angles is provided by all sensors and made available at the clusterhead for deployment-related computations.

(A3) We adopt the discrete coordinate system, in which the monitoring area (sensing field) is represented by a 2D grid network. Locations and orientations of all sensors are obtained via the pre-deployed RFID (Radio-Frequency IDentification) platform or some existing localization technique combined with compass system, and constantly updated to the clusterhead. Neighboring nodes under the adopted coordinate system are defined as sensors within the sensing range (r_s), which is normally much smaller than the radio communication distance (r_c). Although the sensing behavior is directional, we have the omnidirectional communication model in our network. Without loss of generality, we assume that $r_c > 2r_s$ in our model. According to the derivations in [20], [33], if the radio communication range (r_c) is at least twice the sensing radius (r_s), complete coverage of a convex area implies connectivity among the working set of sensor nodes.[5] Consequently, in this work, we only deal with the sensing coverage, and network connectivity follows accordingly.

4 Enhanced Deployment Algorithm (I)

This enhanced deployment algorithm combines virtual forces (VF) and a novel concept of virtual boundary torques to guide sensors movements and rotations respectively. In EDA-I, we propose to create virtual omnidirectional sensors that simulate the actual directional sensors. By mapping distinct combinations of sensing distance and spread angle in directional sensors to dissimilar values of sensing radius in the created virtual omnidirectional sensors, we reduce the originally two-parameter deployment function into a single-parameter operation. Consequently, simple yet effective algorithms can be devised for deploying heterogeneous directional sensors. Two ways of obtaining the corresponding virtual omnidirectional sensors are presented in Section 4.1. Then Section 4.2 details how virtual forces are applied to those virtual sensors for increasing sensing region covered by the actual sensors, along with derivation of suitable threshold for selecting incircles or excircles based on sensor population and fan parameters. In Section 4.3, we introduce a novel concept of virtual boundary torques and illustrate how to exert boundary torques for instructing sensors rotations. Section 4.4 summarizes the EDA-I mechanism.

4.1 Creating Virtual Omnidirectional Sensors

To approximate a sector (fan), we obtain the inscribed circle (internally tangent circle) or the circumscribed circle, as illustrated in Fig. 2. For a directional sensor s_i located at (x_i, y_i) with sensing distance r_i and spread angle δ_i, the obtained inscribed circle (abbreviated as *incircle*) indicates the sensing region of the corresponding virtual omnidirectional sensor s_i^{in} located at (x_i^{in}, y_i^{in}) with sensing radius r_i^{in}. Through simple geometric calculations, we have $(x_i^{in}, y_i^{in}) = (x_i + (r_i - r_i^{in})\cos\theta_i,\ y_i + (r_i - r_i^{in})\sin\theta_i)$, where $r_i^{in} = \frac{r_i \sin(\frac{\delta_i}{2})}{1+\sin(\frac{\delta_i}{2})}$. Likewise, the obtained circumscribed circle (shortened as *excircle*) renders the sensing region of the corresponding virtual omnidirectional sensor s_i^{ex} located at (x_i^{ex}, y_i^{ex}) having sensing radius r_i^{ex}. Based on similar geometric computations, we have $(x_i^{ex}, y_i^{ex}) = (x_i + r_i^{ex}\cos\theta_i,\ y_i + r_i^{ex}\sin\theta_i)$, where $r_i^{ex} = \frac{r_i}{2\cos(\frac{\delta_i}{2})}$. As a result, we construct a *virtual sensors set* $\{s_1', s_2', \ldots, s_k'\}$ for the original sensors set $\{s_1, s_2, \ldots, s_k\}$, given k sensors available for our deployment.[6] Based on whether incircles or excircles are created, the virtual sensors set $\{s_1', s_2', \ldots, s_k'\}$ can be made $\{s_1^{in}, s_2^{in}, \ldots, s_k^{in}\}$ or

4. Although probabilistic sensing models better approach a real sensors system, in our current work, we adopt the binary sensing model in order not to complicate the proposed coverage algorithms.

5. For rotatable directional sensors, the sensing range (coverage) is omnidirectional by rotating viewing angles in all directions. As a result, the derivations developed in [20], [33] can still be utilized to ensure network connectivity as long as the requirement $r_c > 2r_s$ holds.

6. Creating virtual sensors conveniently reduces a two-parameter deployment function into a single-parameter operation, which greatly facilitates the virtual forces calculation presented later in Section 4.2, at the cost of sensing area inaccuracies (gaps), which can be nicely compensated by applying proper attractive/repulsive forces on virtual sensors.

$\{s_1^{ex}, s_2^{ex}, \ldots, s_k^{ex}\}$. Our virtual forces calculations operate on the virtual sensors set directly. Next we describe the designs and actions of virtual forces.

4.2 Applying Virtual Forces

The concept of virtual forces is inspired by the combined idea of potential field and disk packing theory [15], [19]. Each sensor behaves as a source giving a force to others. This force can be either positive (attractive) or negative (repulsive). If two sensors are too close, they exert repulsive forces to separate each other, otherwise they exert attractive forces to draw each other. We quantify the definition of "closeness" by using the distance threshold d_{th}^{ij} for any two sensors s_i and s_j with respective sensing radius r_i and r_j. On the other hand, define parameters w_a and w_r as the weight measurements associated with the attractive and repulsive forces separately. The designs of d_{th}^{ij} and weight constants (w_a, w_r) are critical for the deployment efficacy. Here we adopt the parameter design guidelines provided in our previous publications [17], [18].[7] According to our derivations, the distance threshold should be set on a node-pair basis, and thus $d_{th}^{ij} = \alpha(r_i + r_j)$, where $0 < \alpha \leq 1$ (in our simulations, we set $\alpha = 0.86$). For the weight constants, we proved that good choices for w_a and w_r greatly depend on sensor population and monitored area dimensions, while independent of sensing radius. Thus, given k omnidirectional sensors to be deployed in an $m \times n$ monitoring area, the settings should be $w_a = \frac{1}{k}$ and $w_r = k\sqrt{m^2 + n^2}$. Following the aforementioned parameter settings, we then apply the virtual forces algorithm on those virtual omnidirectional sensors created in Section 4.1. Depending on whether incircles or excircles are obtained, the created virtual sensors set $\{s_1', s_2', \ldots, s_k'\}$ can be $\{s_1^{in}, s_2^{in}, \ldots, s_k^{in}\}$ or $\{s_1^{ex}, s_2^{ex}, \ldots, s_k^{ex}\}$.

4.2.1 Derivation of Suitable Threshold for Creating Incircles or Excircles

We observe that by using incircles, more overlappings are produced among the actual sensors (after applying virtual forces calculations). On the flip side, excircles allow more spacings between sensors. This motivates us to devise an adaptive deployment algorithm that adapts to actual sensor population. When the available sensors are sufficient to fully cover the area, we prefer overlappings between sensors to reduce surveillance holes. In contrast, when sensors are insufficient, more spacings may be beneficial. Given the monitoring area of size $A = m \times n$, and the total area covered by all fan sensors $A_f = \sum_{i=1}^{k} \pi r_i^2(\frac{\delta_i}{2\pi})$, define a factor $\widetilde{a} = \frac{A_f}{A}$, which indicates the maximal possible coverage ratio. Intuitively, the case of having sufficient sensors to fully cover the area occurs when $\widetilde{a} \geq 1$, as shown in Fig. 3(left). However, virtual forces will bring incircles even closer, as illustrated in Fig. 3(right), hence more sensors are required to realize the sufficient condition. For sensor s_i, let A_{f_i} and A_{c_i} denote the areas of the fan-shaped sensor and corresponding incircle respectively, where $A_{f_i} = \pi r_i^2(\frac{\delta_i}{2\pi})$ and $A_{c_i} = \pi r_i^2 \frac{\sin^2\frac{\delta_i}{2}}{(1+\sin\frac{\delta_i}{2})^2}$. Define $\gamma_{f_i} = \frac{A_{f_i} - A_{c_i}}{A_{f_i}}$ to indicate the ratio of non-incircle area inside the fan with respect to the sector area. We can approximate the average non-incircle area ratio $\gamma_f = \frac{\sum_{i=1}^{k} \gamma_{f_i}}{k}$. Suppose the incircles perfectly cover the monitoring region, we require $A_f - \gamma_f \times A_f \geq A$. Thus the sufficient condition occurs when $\frac{A_f}{A} \geq \frac{1}{1-\gamma_f}$, or equivalently $\widetilde{a} \geq \frac{1}{1-\gamma_f}$. Consequently, a suitable threshold for selecting incircles or excircles can be made by averaging[8] the two sufficient conditions to obtain $\widetilde{a} \geq \frac{2-\gamma_f}{2(1-\gamma_f)}$. Define γ_{thre} as $\frac{2-\gamma_f}{2(1-\gamma_f)}$. We propose to create incircles when $\widetilde{a} \geq \gamma_{thre}$, and use excircles otherwise.[9]

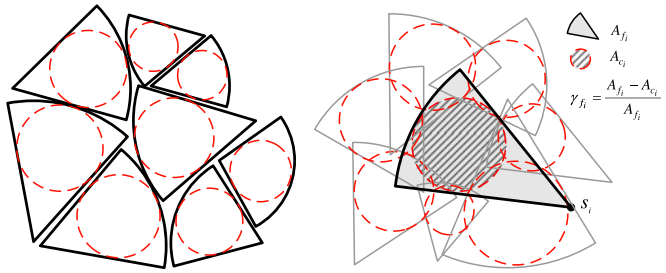


Fig. 3. Illustration of sufficient conditions when $\widetilde{a} \geq 1$ (left) and when $\widetilde{a} \geq \frac{1}{1-\gamma_f}$ (right).

4.3 New Concept of Virtual Boundary Torques (VBT)

After re-positioning sensors, we observe that the sensing coverage can be further enhanced by swiveling sensors to face toward boundaries.[10] While the use of virtual forces can guide sensors to move, it is incapable of manipulating sensors viewing (facing) angles. Therefore, in this section, we introduce a novel concept of virtual torques that are given by the boundaries to guide sensors rotations.

A torque[11] for sensor s_i is basically the amount of rotating angle centered at location (x_i, y_i). The rotating angle can be counterclockwise or clockwise. In this paper, we define the counterclockwise rotation as positive (+) and clockwise negative (−). Virtual boundary torques are virtual torques received by a sensor from the boundaries. When there is

7. The results suggest that parameters d_{th}^{ij}, w_a and w_r play important roles in determining the achievable coverage ratios, while the overlapping factor α has relatively insignificant impact on coverage performance. More details can be found in [17], [18].

8. Since the incircles cannot perfectly cover the monitoring area, we propose to use a less strict threshold by averaging the two sufficient conditions. From our experiments, an averaged threshold does produce better coverage performance than using the larger (higher) or the smaller (lower) threshold of the two conditions.

9. Preliminary developments of a self-deployment protocol, called OPT, combining virtual forces and a novel concept of boundary torques, can be found in our previous publication [25]. In this paper, EDA-I is an improved version of OPT for better sensing coverage achievements in a bounded monitoring area. The main operational difference between OPT and EDA-I lies in the criterion ($\widetilde{a}$) for selecting incircles or excircles. Specifically, OPT simply sets the threshold $\gamma_{thre} = 1$, while EDA-I proposes a more sophisticated threshold to be used by setting $\gamma_{thre} = \frac{2-\gamma_f}{2(1-\gamma_f)}$ as derived here.

10. With sensors facing outwards (toward boundaries), events arising inside the bounded region can still be successfully detected as long as the corresponding locations are covered by the visual (directional) sensors.

11. A torque is a force multiplied by distance (in an angular direction). Here we quantify the torque by the amount of rotating angle.

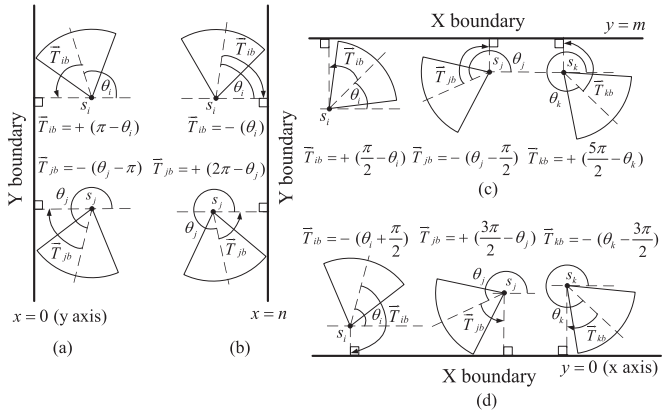


Fig. 4. Computations of virtual boundary torques for boundaries along (a) *y* axis, (b) $x = n$, (c) $y = m$, and (d) *x* axis, respectively in a 2D bounded monitoring area.

only one boundary, the virtual torque is given such that a sensor can alter its viewing direction toward the boundary. In our coordinate system, illustrated in Fig. 4, there are two types of Y boundaries, including $x = 0$ and $x = n$, and two kinds of X boundaries, containing $y = 0$ and $y = m$. Below we discuss how to model the virtual boundary torques under the four conditions.

Define the virtual boundary torque given to sensor s_i as $\overrightarrow{T}_{ib}$. When we consider the boundary of $x = 0$, as shown in Fig. 4a, the virtual boundary torque $\overrightarrow{T}_{ib}$ for sensor s_i with viewing angle θ_i no greater than π can be obtained as $+(\pi - \theta_i)$, which directs s_i to revolve counterclockwise for $(\pi - \theta_i)$ degrees. On the other hand, for sensor s_j with viewing angle θ_j greater than π, the virtual boundary torque $\overrightarrow{T}_{jb}$ can be computed as $-(\theta_j - \pi)$, indicating s_j should rotate clockwise for $(\theta_j - \pi)$ degrees. The purpose of separating the two cases is to limit each rotation angle within π (half circle), so that rotations energy can be reasonably conserved. Similar design principles are applied to other boundary conditions. For the boundary of $x = n$, two cases are considered, while three cases should be modeled separately for both X boundaries ($y = 0$ and $y = m$)[12]. Below we summarize all cases for computing the virtual boundary torques $\overrightarrow{T}_{ib}$, where b stands for "boundary".

For sensor s_i having the boundary of $x = 0$ (y axis), as shown in Fig. 4a, we have

$$\overrightarrow{T}_{ib} = \left\{ \begin{array}{ll} +(\pi - \theta_i) & \text{if } 0 \le \theta_i \le \pi \\ -(\theta_i - \pi) & \text{otherwise} \end{array} \right\}. \quad (1)$$

For sensor s_i having the boundary of $x = n$, as shown in Fig. 4b, the virtual boundary torque $\overrightarrow{T}_{ib}$ is modeled as

$$\overrightarrow{T}_{ib} = \left\{ \begin{array}{ll} -(\theta_i) & \text{if } 0 \le \theta_i \le \pi \\ +(2\pi - \theta_i) & \text{otherwise} \end{array} \right\}. \quad (2)$$

12. The Y boundaries only have two cases in regards with the orientation to the horizontal line ($y = y_i$) in determining the viewing angles, to limit each rotation degree (value) within a half circle (π). On the other hand, the X boundaries have three cases because of the wider possibility of the viewing angles toward the X boundaries, aiming to keep the rotation degree a positive value.

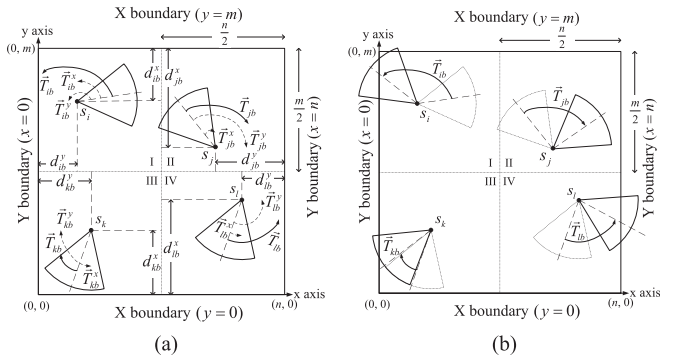


Fig. 5. Illustration of (a) resultant virtual boundary torques and (b) virtual rotations applied to respective sensor node.

For sensor s_i having the boundary of $y = m$, as shown in Fig. 4c, we have virtual boundary torque

$$\overrightarrow{T}_{ib} = \left\{ \begin{array}{ll} +(\frac{\pi}{2} - \theta_i) & \text{if } 0 \le \theta_i \le \frac{\pi}{2} \\ -(\theta_i - \frac{\pi}{2}) & \text{if } \frac{\pi}{2} < \theta_i \le \frac{3\pi}{2} \\ +(\frac{5\pi}{2} - \theta_i) & \text{otherwise} \end{array} \right\}. \quad (3)$$

For sensor s_i having the boundary of $y = 0$ (x axis), as shown in Fig. 4d, we obtain the virtual boundary torque $\overrightarrow{T}_{ib}$ as

$$\overrightarrow{T}_{ib} = \left\{ \begin{array}{ll} -(\theta_i + \frac{\pi}{2}) & \text{if } 0 \le \theta_i \le \frac{\pi}{2} \\ +(\frac{3\pi}{2} - \theta_i) & \text{if } \frac{\pi}{2} < \theta_i \le \frac{3\pi}{2} \\ -(\theta_i - \frac{3\pi}{2}) & \text{otherwise} \end{array} \right\}. \quad (4)$$

Next, we extend the concept to consider virtual torques from multiple boundaries. In a 2D rectangular region, we define the virtual boundary torque imposed on a sensor as *a combined torque from the nearest X boundary and nearest Y boundary*. Depending on the coordinate of sensor s_i, our Q function first determines which quadrant s_i resides in, where

$$Q(x_i, y_i) = \left\{ \begin{array}{ll} \text{I} & \text{if } x_i \le \frac{n}{2} \&\& y_i > \frac{m}{2} \\ \text{II} & \text{if } x_i > \frac{n}{2} \&\& y_i > \frac{m}{2} \\ \text{III} & \text{if } x_i \le \frac{n}{2} \&\& y_i \le \frac{m}{2} \\ \text{IV} & \text{if } x_i > \frac{n}{2} \&\& y_i \le \frac{m}{2} \end{array} \right\}. \quad (5)$$

As illustrated in Fig. 5a, for sensors located in quadrant I, the boundaries of $x = 0$ and $y = m$ need be considered. Similarly, for sensors falling in quadrants II, III, and IV, one nearest X boundary and one nearest Y boundary are considered to produce a combined virtual boundary torque. Take sensor s_i in Fig. 5a for example, denote the virtual torque from X boundary as $\overrightarrow{T}^x_{ib}$ and torque from Y boundary as $\overrightarrow{T}^y_{ib}$, the resultant virtual boundary torque exerted on s_i is a combined torque of $\overrightarrow{T}^x_{ib}$ and $\overrightarrow{T}^y_{ib}$. Instead of combining the two torques with equal importance, we define a weight parameter $\beta = \frac{d^x_{ib}}{d^y_{ib}}$, where d^x_{ib} and d^y_{ib} denote the euclidean distances between s_i and the nearest X and Y boundaries respectively. The weight parameter β is then used to give the closer boundary more impact on the resultant torque computation by defining $\overrightarrow{T}_{ib} = \frac{\overrightarrow{T}^x_{ib} + \beta \overrightarrow{T}^y_{ib}}{1+\beta}$. Once the resultant virtual torques are obtained, sensors perform virtual rotations, as illustrated in Fig. 5b. Some of the sensors in this figure may

seem to render wasted coverage outside the boundaries after rotations. Nevertheless, in our EDA-I, an iteration actually includes both the exertions of virtual forces and virtual boundary torques. Therefore, those sensors have good chance to be moved inward in the subsequent iterations.

4.4 EDA-I Summary

Algorithm 1 provides the pseudocode for EDA-I deployment mechanism. The algorithm terminates when either the required sensing coverage threshold (c_{th}) or the maximum allowable number of iterations ($Maxloops$) is reached.[13] Note that the forces calculations operate on virtual sensors, while boundary torques computations work on actual sensors. Both virtual movements and rotations exert on actual sensors. In the end of each loop (iteration), every sensor performs virtual movement and rotation without physically moving and rotating themselves. Physical movements and rotations are conducted once the deployment calculation process terminates.

Algorithm 1. Enhanced Deployment Algorithm (I)

1: create corresponding *virtual omnidirectional sensors* $\{s_1^{'}, s_2^{'}, \ldots, s_k^{'}\}$ for all directional sensors $\{s_1, s_2, \ldots, s_k\}$ by the following rule;
2: **if** $\tilde{a} \geq \gamma_{thre}$ **then**
3: obtain *virtual omnidirectional sensors* $\{s_1^{'}, s_2^{'}, \ldots, s_k^{'}\}$ as $\{s_1^{in}, s_2^{in}, \ldots, s_k^{in}\}$; // incircles
4: **else**
5: obtain *virtual omnidirectional sensors* $\{s_1^{'}, s_2^{'}, \ldots, s_k^{'}\}$ as $\{s_1^{ex}, s_2^{ex}, \ldots, s_k^{ex}\}$; // excircles
6: set $loops = 0$;
7: set $c_{now} = c_{init}$; // initial coverage ratio
8: **while** ($loops < Maxloops$) && ($c_{now} < c_{th}$) **do**
9: **for** each virtual sensor $s_i^{'} \in \{s_1^{'}, s_2^{'}, \ldots, s_k^{'}\}$ **do**
10: compute $\overrightarrow{F}_i^{'} = \sum_{j \neq i, j=1}^{k} \overrightarrow{F}_{ij}^{'} + \overrightarrow{F}_{ib}^{'}$;
11: perform *virtual movements* on all sensors;
12: **for** each actual sensor $s_i \in \{s_1, s_2, \ldots, s_k\}$ **do**
13: compute virtual boundary torque $\overrightarrow{T}_{ib}$;
14: perform *virtual rotations* on all sensors;
15: update *coverage ratio* c_{now};
16: set $loops = loops + 1$;

5 Enhanced Deployment Algorithm (II)

In this section, we propose another deployment algorithm, EDA-II, to enhance the sensing coverage by applying Voronoi diagram instructed movements and boundary torques guided rotations on directional sensors. We provide preparatory background of the Voronoi diagram data structure and introduce a simple yet efficient technique to compute a Voronoi diagram for a given discrete set of point sites in Section 5.1. Then Section 5.2 details how to perform sensors movements directed by the constructed Voronoi diagram, while Section 5.3 presents how VBT (virtual boundary torques) technique should be adapted to employ appropriate rotations on directional sensors. We wrap up the EDA-II mechanism in Section 5.4.

13. If later the coverage ratio goes below a certain threshold due to faulty sensors, EDA-I will be invoked to perform a global redeployment.

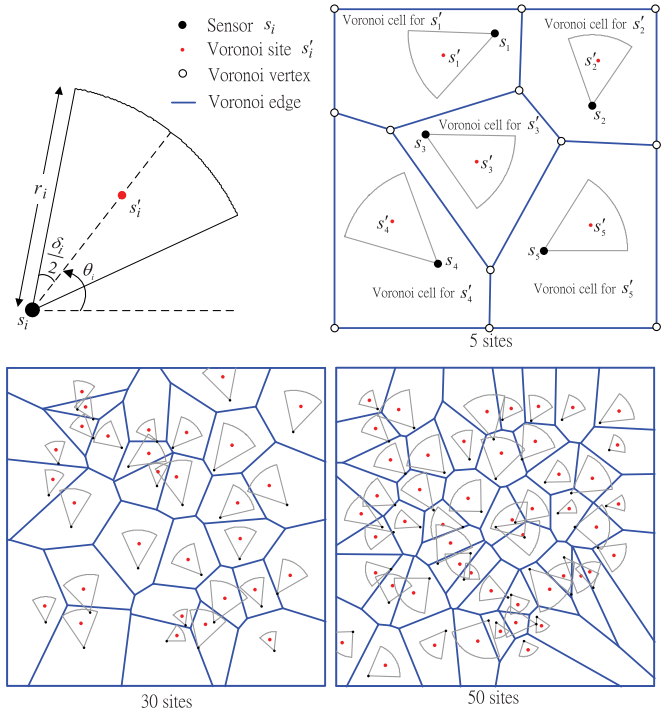


Fig. 6. Display of constructed Voronoi diagrams for 5, 30, and 50 point sites, respectively.

5.1 Voronoi Diagram Preliminaries

For any directional sensor s_i located at (x_i, y_i) with sensing distance r_i and spread angle δ_i, EDA-II approximates the sensing sector (fan) by creating a virtual sensor $s_i^{'}$ located at the centroid point $(x_i^{'}, y_i^{'})$ of the fan, where $(x_i^{'}, y_i^{'}) = (x_i + \frac{2}{3} r_i \frac{\sin\frac{\delta_i}{2}}{\frac{\delta_i}{2}} \cos\theta_i,\ y_i + \frac{2}{3} r_i \frac{\sin\frac{\delta_i}{2}}{\frac{\delta_i}{2}} \sin\theta_i)$. Let created virtual sensors represent corresponding Voronoi point sites. Voronoi diagrams are a fundamental data structure in computational geometry, which is involved with solving geometrical problems. Given a discrete set of point sites, a Voronoi diagram divides the plane into regions, called Voronoi cells or polygons, each associated with a site. The Voronoi cell of a site $s_i^{'}$ contains the set of points in the plane for which $s_i^{'}$ is the closest site among all the sites. One direct application of a Voronoi diagram is the nearest query problem. A comprehensive survey of Voronoi diagrams and related structures, with particular emphasis on the unified exposition of their mathematical and computational properties can be found in [9]. Fig. 6 illustrates the Voronoi diagrams for given 5, 30, and 50 sites (virtual sensors) respectively.

To construct a Voronoi diagram, we adopt the sweepline algorithm proposed in [12], which runs in $O(k \log k)$ worst-case computation time for k given point sites. As shown in Fig. 7 (upper left), the algorithm views each site (virtual sensor) as the *focus* of a parabola and utilizes a sweepline L to act as a moving *directrix*. While the horizontal line L (directrix) sweeps downward across the plane, the parabolas corresponding to traversed sites grow accordingly. By recording the trace of parabola intersections, a Voronoi edge can be generated. To observe this, suppose two sites (F_a and F_b) are given, below we prove that the trace of parabola intersections forms a Voronoi edge for the two sites after performing the sweepline algorithm.

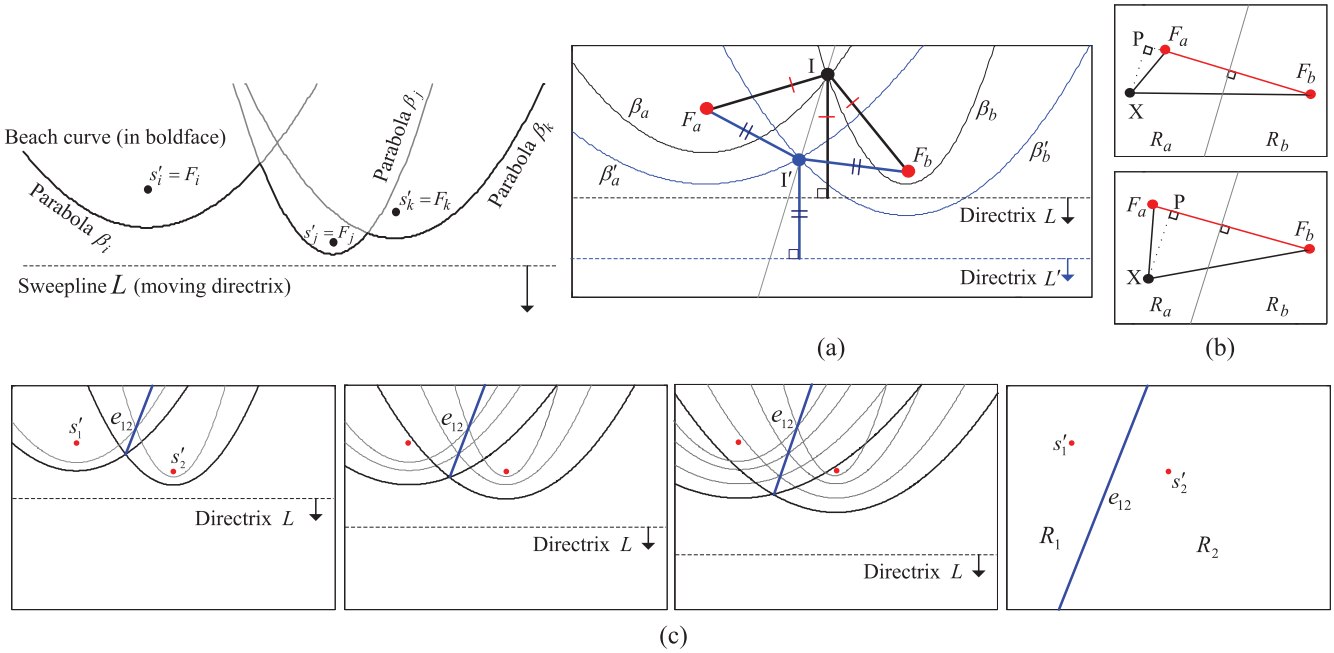


Fig. 7. Basic operations of the sweepline algorithm and process of constructing a two-site Voronoi diagram.

Lemma 1. *Given two sites, F_a and F_b, the trace of parabola intersections created by the sweepline algorithm forms a perpendicular bisector of line segment $\overline{F_aF_b}$.*

Proof. Suppose I is the parabola intersection with directrix L, whereas I' is the parabola intersection when the sweepline moves to the location of directrix L', as shown in Fig. 7a. Since I resides on both parabolas β_a and β_b, by definition, we have $|\overline{IF_a}| = d(I, L)$ and $|\overline{IF_b}| = d(I, L)$, where $d(I, L)$ indicates the perpendicular distance from point I to line L. That is, $|\overline{IF_a}| = |\overline{IF_b}|$. Similarly, we have $|\overline{I'F_a}| = |\overline{I'F_b}|$. In other words, as the sweepline (moving directrix) scans downward, any point on the trace of parabola intersections keeps equal distance from site F_a to that from site F_b. Consequently, the trace of parabola intersections forms a perpendicular bisector of line segment $\overline{F_aF_b}$. □

Lemma 2. *A perpendicular bisector of line segment $\overline{F_aF_b}$ partitions the plane into two regions, R_a (containing site F_a) and R_b (containing site F_b), such that any point in R_a is closer to site F_a than to site F_b, and vice versa.*

Proof. Suppose X is a point in R_a and P is the projection of X on the extended line of $\overline{F_aF_b}$, as shown in Fig. 7b. Two cases are considered. If P is outside of line segment $\overline{F_aF_b}$, then $|\overline{F_aP}| < |\overline{F_bP}|$, which implies $|\overline{XF_a}| < |\overline{XF_b}|$ according to the Pythagorean theorem. On the other hand, if P is located on line segment $\overline{F_aF_b}$, then $|\overline{F_aP}| < |\overline{F_bP}|$ since P is guaranteed to reside on the left of the perpendicular bisector, as plotted in Fig. 7b (lower), leading to $|\overline{XF_a}| < |\overline{XF_b}|$. As a result, any point in R_a is closer to site F_a than to site F_b. Similarly, any point in R_b is closer to site F_b than to site F_a. □

Theorem 1. *Given two sites, F_a and F_b, the trace of parabola intersections created by the sweepline algorithm forms a Voronoi edge for sites F_a and F_b.*

Proof. By Lemma 1 and Lemma 2, the trace of parabola intersections created by the sweepline algorithm is exactly a Voronoi edge, which divides the plane into two Voronoi cells (polygons). □

Fig. 7c shows the process of constructing a two-site Voronoi diagram, while Fig. 8 demonstrates a more complex example of constructing a Voronoi diagram for five sites. The sweepline algorithm is an efficient and reasonably simple technique for computing Voronoi diagrams. Voronoi diagrams have proven to be a powerful tool in solving seemingly unrelated computational problems. In EDA-II, we leverage the constructed Voronoi diagram to guide our sensors movements.

5.2 Voronoi Diagram Directed Movements

For each Voronoi cell (polygon), EDA-II examines whether the corresponding sensing fan covers all Voronoi vertices surrounding the polygon. If not all vertices are covered, the corresponding virtual sensor will be moved to the centroid location of the Voronoi polygon. Otherwise, the sensor location remains unchanged.[14] Suppose a Voronoi site (virtual sensor) s'_i is positioned at (x'_i, y'_i) and the centroid location of corresponding Voronoi polygon is (x_{c_i}, y_{c_i}). Assume the Voronoi polygon consists of n vertices, denoted as $v_i^1, v_i^2, \ldots, v_i^n$, located at $(x_{v_i^1}, y_{v_i^1})$, $(x_{v_i^2}, y_{v_i^2})$, $\ldots$, $(x_{v_i^n}, y_{v_i^n})$ respectively, as illustrated in Fig. 9. To formulate the EDA-II examination process, define a boolean function as Eq. (6) which indicates whether or not the sensing fan covers vertex v_i^h, $\forall h = 1, 2, \ldots, n$, where ϕ_i^h is the angle of line $\overrightarrow{s_iv_i^h}$ with respect to the horizontal line of $y = y_i$ in counterclockwise direction, as shown in Fig. 9 (left).

14. By setting a strict condition for not-moving, we intend to raise the moving probability, so that the fan centroid (virtual sensor) tends to move toward the corresponding Voronoi cell's centroid, thus increasing the covered region.

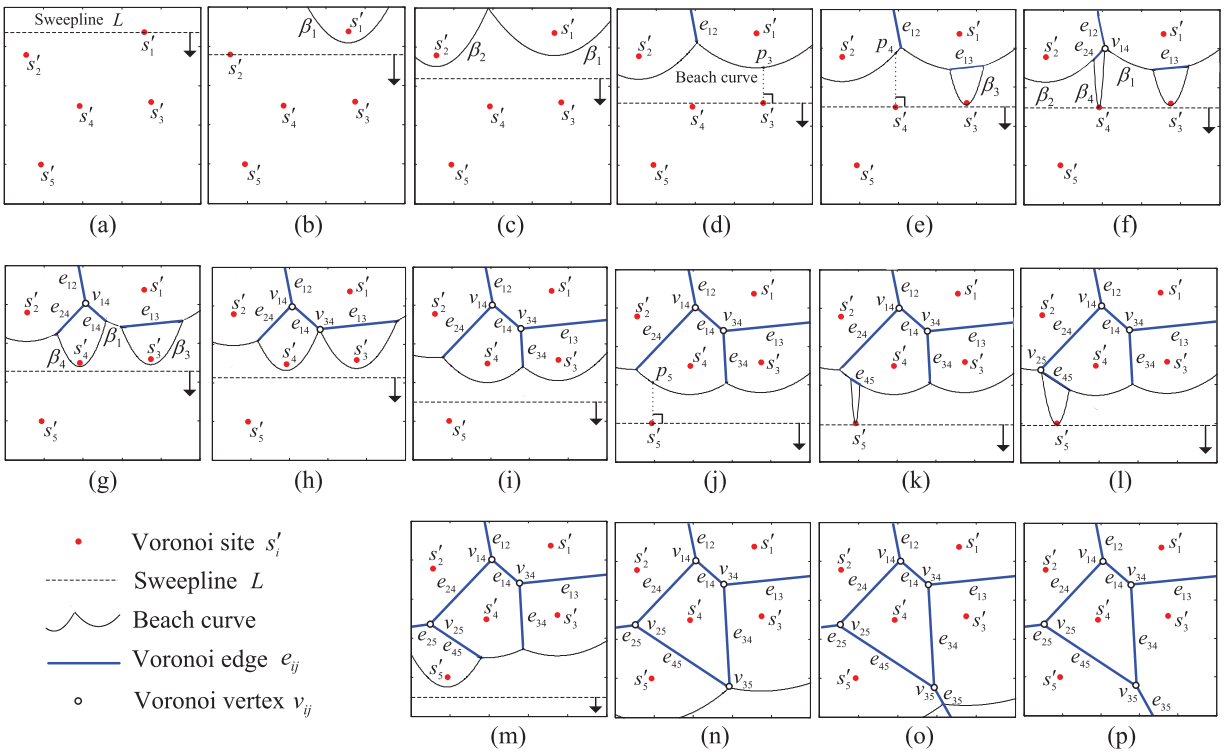


Fig. 8. Example of constructing a Voronoi diagram for five sites (virtual sensors).

Then the new location of sensor s_i can be determined as Eq. (7),

$$D(v_i^h) = \left\{ \begin{array}{ll} 1 & \text{if } |\overrightarrow{s_i v_i^h}| \le r_i \;\&\&\; (\theta_i - \frac{\delta_i}{2} \le \phi_i^h \le \theta_i + \frac{\delta_i}{2} \parallel \theta_i - \frac{\delta_i}{2} \le \phi_i^h + 2\pi \le \theta_i + \frac{\delta_i}{2}) \\ 0 & \text{otherwise} \end{array} \right\}. \tag{6}$$

$$\text{new } (x_i,\ y_i) = \left\{ \begin{array}{ll} (x_i,\ y_i) & \text{if } \prod_{h=1}^{n} D(v_i^h) = 1 \\ (x_i + x_{c_i} - x_i^{'},\ y_i + y_{c_i} - y_i^{'}) & \text{if } \prod_{h=1}^{n} D(v_i^h) = 0 \end{array} \right\} \tag{7}$$

$$\begin{aligned} x_{c_i} &= \frac{1}{6A_{p_i}} \sum_{h=0}^{n-1} (x_{v_i^{(h \bmod n)+1}} + x_{v_i^{[(h+1) \bmod n]+1}})(x_{v_i^{(h \bmod n)+1}} y_{v_i^{[(h+1) \bmod n]+1}} - x_{v_i^{[(h+1) \bmod n]+1}} y_{v_i^{(h \bmod n)+1}}) \\ y_{c_i} &= \frac{1}{6A_{p_i}} \sum_{h=0}^{n-1} (y_{v_i^{(h \bmod n)+1}} + y_{v_i^{[(h+1) \bmod n]+1}})(x_{v_i^{(h \bmod n)+1}} y_{v_i^{[(h+1) \bmod n]+1}} - x_{v_i^{[(h+1) \bmod n]+1}} y_{v_i^{(h \bmod n)+1}}) \end{aligned} \tag{8}$$

where $(x_i^{'},\ y_i^{'}) = (x_i + \frac{2}{3} r_i \frac{\sin\frac{\delta_i}{2}}{\frac{\delta_i}{2}} \cos\theta_i,\ y_i + \frac{2}{3} r_i \frac{\sin\frac{\delta_i}{2}}{\frac{\delta_i}{2}} \sin\theta_i)$ and $A_{p_i} = \frac{1}{2}\sum_{h=0}^{n-1}(x_{v_i^{(h \bmod n)+1}} y_{v_i^{[(h+1) \bmod n]+1}} - x_{v_i^{[(h+1) \bmod n]+1}} y_{v_i^{(h \bmod n)+1}})$, representing the area size of the Voronoi polygon. Fig. 9 (right) displays the result of moving sensor s_i to the new location.

5.3 Adapted VBT

In EDA-II, some sensors are likely to be moved outside the boundary after performing Voronoi diagram directed movements. To remedy this problem and ensure our VBT operates correctly, we propose to *flip* sensors inside before executing the virtual boundary torques calculations. Specifically, for sensor s_i located at $(x_i,\ y_i)$ outside of boundary, EDA-II simply alters its viewing angle for 180 degrees to face the opposite side centered at the location of virtual sensor $s_i^{'}$, where new $(x_i,\ y_i) = (2x_i^{'} - x_i,\ 2y_i^{'} - y_i)$. Fig. 10 shows the process of flipping an outside sensor inside the boundary. Once all outside sensors are fixed, the VBT

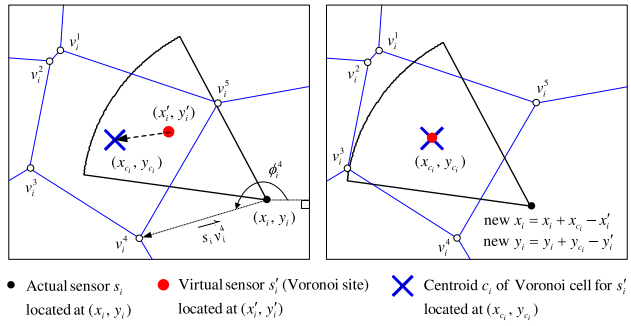


Fig. 9. Sensor movement guided by the computed Voronoi diagram.

procedure as presented in Section 4.3 comes into play for guiding sensors rotations.

5.4 EDA-II Summary

Algorithm 2 provides the pseudocode for our EDA-II deployment mechanism. The algorithm terminates when either the required sensing coverage threshold (c_{th}) or the maximum allowable number of iterations ($Maxloops$) is reached.[15] Note that the Voronoi diagram is constructed for virtual sensors (Voronoi sites), while boundary torques computations work on actual sensors. Both virtual movements and rotations exert on actual sensors. In the end of each loop (iteration), every sensor performs virtual movement and rotation without physically moving and rotating themselves. Physical movements and rotations are conducted once the deployment calculation process terminates.

Algorithm 2. Enhanced Deployment Algorithm (II)

1: create corresponding *virtual sensors* $\{s_1^{'}, s_2^{'}, \ldots, s_k^{'}\}$ for all directional sensors $\{s_1, s_2, \ldots, s_k\}$ by computing centroid points of all sensing sectors;
2: set $loops = 0$;
3: set $c_{now} = c_{init}$; // initial coverage ratio
4: **while** ($loops < Maxloops$) && ($c_{now} < c_{th}$) **do**
5: construct a corresponding Voronoi diagram for the current locations (sites) of all virtual sensors;
6: perform Voronoi diagram directed *virtual movements* on all sensors;
7: **for** each actual sensor $s_i \in \{s_1, s_2, \ldots, s_k\}$ **do**
8: **if** (s_i is located outside boundary) **then**
9: new $(x_i,\ y_i) = (2x_i^{'} - x_i,\ 2y_i^{'} - y_i)$; // sensor flips inside boundary
10: compute virtual boundary torque $\overrightarrow{T}_{ib}$;
11: perform *virtual rotations* on all sensors;
12: update *coverage ratio* c_{now};
13: set $loops = loops + 1$;

6 Performance Evaluation

6.1 Achievable Coverage Ratio, Energy Consumption, and Computation Time

We simulate heterogeneous sensors by setting the sensing radius uniformly distributed in $[45, 72]$ and spread angle uniformly distributed in $[\frac{\pi}{4}, \frac{\pi}{3}]$[16]. Abbreviation HRA is used to indicate the settings for heterogeneous sensors with Heterogeneous Sensing Radius and Spread Angle. In addition to EDA-I and EDA-II, two existing coverage-enhancing algorithms are implemented: Sensing Connected Sub-Graph (SCSG) [28] and Virtual Centripetal Force (VCF) [37]. The SCSG approach divides a directional sensing network into several sensing connected sub-graphs. For each group of sensors (sensing connected sub-graph), a convex hull is constructed. Then SCSG basically instructs each sensor on the convex hull to rotate the sensing direction outward (with respect to the convex hull). The VCF method is based on the concept of virtual centripetal forces, which builds grids in the

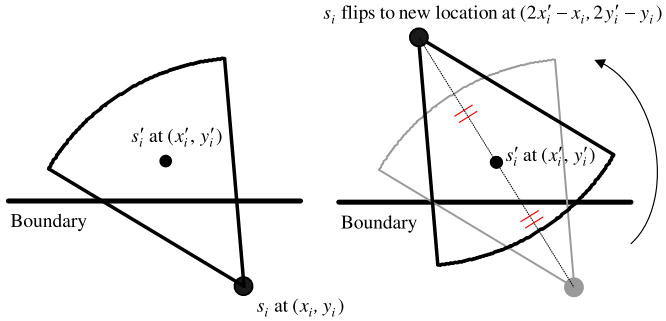


Fig. 10. Sensors outside the boundary caused by performing Voronoi diagram directed movements should flip inside before VBT is applied.

sensing sector for every sensor and defines the magnitude of repel forces from neighboring sensors as the measure of overlapped grid cells. Then VCF computes the resultant force (forces composition) to exert on the centroid point of each sensor's sensing sector, which guides the direction (clockwise or counterclockwise) and amount of degrees (rotation measure) a sensor should employ on its rotation. Since only rotations are performed, the deployment mechanisms proposed by [28], [37] do not perform well in our simulated environments. In addition, their mechanisms consider only homogeneous directional sensors. Thus, in this section, we augment the SCSG and VCF mechanisms by incorporating our virtual forces movement strategy to cooperate with their rotation methods, entitled VF+SCSG and VF+VCF, and adapt the two approaches to work for HRA environments. Six deployment algorithms are implemented: EDA-I, EDA-II, VF+SCSG, VF+VCF, SCSG, and VCF. All six algorithms use $Maxloops = 1000$ and $c_{th} = 0.98$ as their termination conditions.

Fig. 11 displays the deployment results produced by the six mechanisms with 50 and 90 heterogeneous directional sensors. Our EDA-I and EDA-II are capable of achieving a higher sensing coverage than the other strategies. Without movements, SCSG and VCF are not effective in enhancing coverage ratio by rotating sensors at fixed locations. When $k = 50$, the SCSG approach even brings the coverage ratio lower than that of the initial deployment, due to an unnecessary sensing hole (void) caused by forcing sensors to face outward at fixed positions. Also adopting the SCSG rotation strategy, VF+SCSG, on the other hand, improves the sensing coverage noticeably as a result of incorporating the VF movement mechanism. By wisely combining the two actions (movements and rotations) in the deployment function, EDA-I effectively improves the sensing coverage ratio by 37 percent (when $k = 50$) and 40 percent (when $k = 90$), while EDA-II enhances the sensing performance by 44 percent for both cases.

In Fig. 12, we observe the obtained coverage ratio and energy consumption required by respective deployment strategy.[17] The energy consumed by EDA-I, EDA-II, VF+SCSG, and VF+VCF, includes both the physical moving and rotating energy, while SCSG and VCF only consume

15. If later the coverage ratio goes below a certain threshold due to faulty sensors, EDA-II will be invoked to perform a global redeployment.

16. Here, we adopt the standard settings for commodity camera sensors in our simulations to better approach realistic scenarios [1], [2], [7].

17. To address the statistical uncertainties, each data value in our simulation figures has been obtained from an average of 20 experiments with a 95 percent confidence level. Theoretically the maximum (optimal) coverage ratio is set as $\frac{A_f}{A}$, where A denotes the monitored area size and $A_f = \sum_{i=1}^{k} \pi r_i^2 (\frac{\delta_i}{2\pi})$.

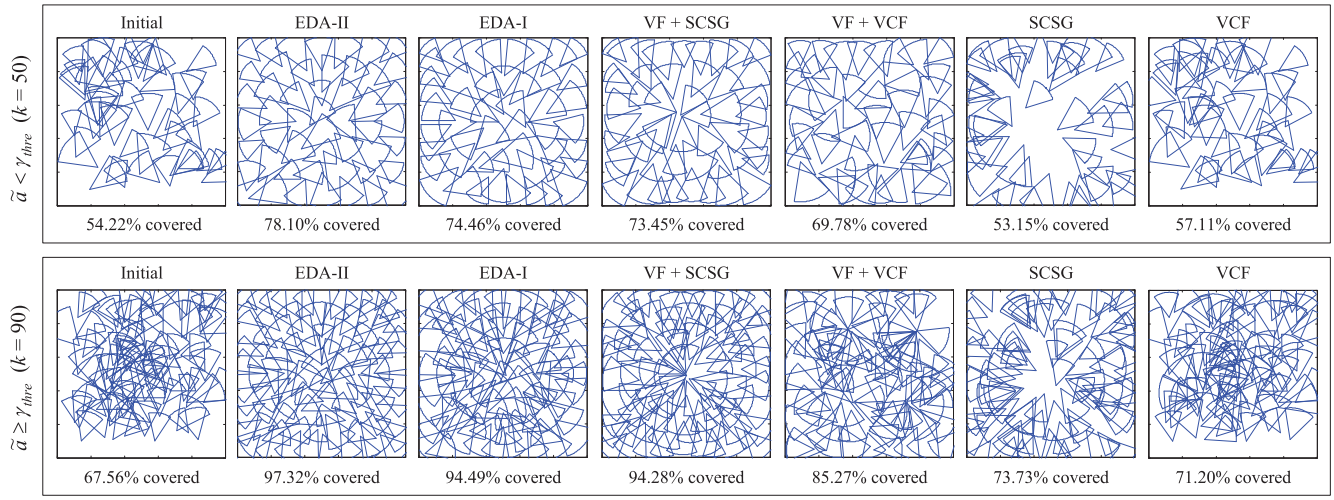


Fig. 11. Sensors distribution status after applying EDA-II, EDA-I, VF+SCSG, VF+VCF, SCSG, and VCF deployment strategies in a 300×300 area with HRA settings for 50 (upper) and 90 (lower) sensor nodes, respectively.

physical rotating energy. To model the energy, we do real experiments on the sensor robot used in our prototyping testbed[18] with grid size equal to 1 cm. The robot assembles six 1.2 V 2000 mAh rechargeable NiMH batteries with measured $200 \sim 290$ mA moving current and average moving speed at 0.06 m/sec. Consequently, the average moving energy consumption per grid (unit distance) can be estimated at $0.29 \times 7.2 \times (\frac{0.01}{0.06}) = 0.348$ Joule. On the other hand, $280 - 310$ mA rotating current and average rotating speed at 96 degree/sec are measured on the same robot device. Thus energy consumption per degree can be estimated at $0.31 \times 7.2 \times (\frac{1}{96}) = 0.02325$ Joule. As shown in Fig. 12, since EDA-I and EDA-II have the adaptiveness to sensor population, higher sensing coverage can always be achieved no matter how many sensors are available for deployment. One interesting result exhibited in this figure is that EDA mechanisms do not consume the most energy among all strategies, despite having the best sensing coverage. In most cases, EDA-II achieves a higher sensing ratio than EDA-I at the expense of more consumed energy. This phenomenon reveals that the Voronoi diagram directed movements adopted by EDA-II are more effective in dispersing sensors to proper monitoring positions. For virtual forces movements utilized by EDA-I, certain imprecisions are likely to occur during the processes of creating virtual omnidirectional sensors and virtual forces calculations. Nonetheless, we demonstrate that our EDA deployment mechanisms can be coverage-effective with moderate consumed energy.

We also investigate the computation time and executed algorithm loops against different required coverage thresholds under respective deployment strategies in Fig. 13. We simulate 70 directional sensors with HRA settings randomly distributed in a 300×300 monitoring area, and vary c_{th} (required coverage threshold) from 0.5 up to 0.9 to observe the used computation time and number of iterations executed by respective strategies. The computation time is measured by running a computer with AMD Athlon64x2 3800+ dual-core processor (at 2 GHz clock rate) and g++ compiler (version 4.3.2) installed on a Linux 2.6.27.5-117 kernel platform as our clusterhead for deployment algorithm computations. As shown in Fig. 13, when $c_{th} = 0.5$, EDA-I spends the least amount of time, given that only one loop is necessary for all four strategies to achieve the required coverage ratio. However, as c_{th} increases, the number of executed loops demanded by EDA-I grows faster than EDA-II and even VF+SCSG or VF +VCF in some cases, leading to extended total computation time consumed by EDA-I. In all cases other than $c_{th} = 0.5$, our EDA-II consumes the least amount of computation time. For achievable sensing coverage within $Maxloops$ iterations, we observe that VF+VCF fails to produce meaningful deployment when $c_{th} > 0.8$, whereas EDA-I and VF+SCSG are incapable of deploying sensors when $c_{th} > 0.85$. For required coverage threshold beyond 0.85, only EDA-II can survive to produce useful

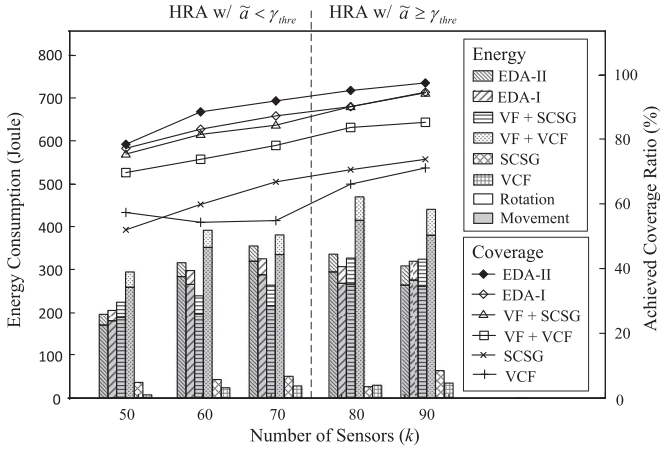


Fig. 12. Coverage ratios attained and energy consumed by EDA-II, EDA-I, VF+SCSG, VF+VCF, SCSG, and VCF mechanisms under various numbers of sensor nodes in a monitored 300×300 area with HRA settings.

18. The mobile sensor used in our testbed is a moving robot (LEGO MINDSTORMS NXT 9797 [4]) carrying a single-board computer (Crossbow Stargate [3]) for computation, a sensor-equipped mote (Crossbow MICAz [6]) for communication, and a webcam (directional visual sensing) device (Logitech QuickCam Pro 4000 [5]).

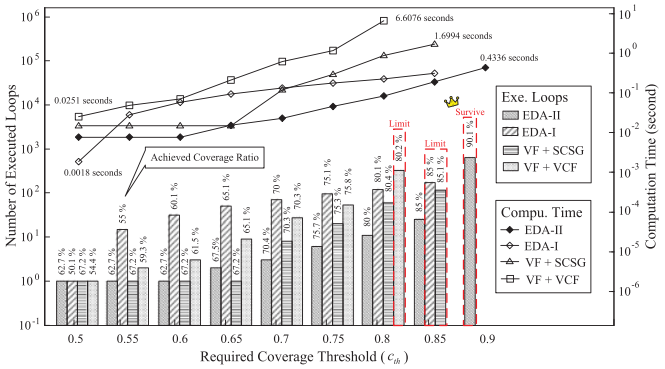


Fig. 13. Comparison of consumed computation time and number of executed loops carried out by respective deployment strategies under certain required coverage thresholds in a 300×300 area ($k = 70$ with HRA settings).

deployment result. From this figure, EDA-II further demonstrates its potential to sustain the highest sensing coverage requirement under reasonable computation time.

6.2 Comparison of Distributed Deployment Algorithms Leveraging Both Movements and Rotations

While EDA-II only allows centralized calculations, our EDA-I has the flexibility of being adapted into a distributed deployment algorithm without requiring global information to translocate sensors. In this section, we adapt the EDA-I mechanism to operate in a distributed manner by collecting locations info from 2-hop neighbors (rather than global info of all sensors locations) for computing virtual forces. Two existing distributed deployment algorithms that also employ both movements and rotations are implemented for comparison purposes: Mobility-Assisted Mobility (MAM) [14] and Distributed Voronoi-based Self-redeployment Algorithm (DVSA) [27] as reviewed in Section 2. Unlike our EDA-I that performs movements *before* rotations in each iteration, the MAM approach exercises sensors movements *after* sensors rotations in a cascaded manner. For DVSA, this approach guides sensors to new locations and new sensing directions by utilizing the features of constructed Voronoi polygons/cells without global information. We use the same environmental settings from Section 6.1 by configuring the sensing radius uniformly distributed in $[45, 72]$ and spread angle uniformly distributed in $[\frac{\pi}{4}, \frac{\pi}{3}]$ in a bounded 300×300 monitoring area. In addition to a random (evenly-placed) sensors initial configuration, we also generate another corner (unevenly-placed) initial configuration. In a monitoring environment where network administrators have little control over the exact sensors deployment, initially concentrated placement of sensor nodes in a corner can serve as a convenient and practical way of starting node configurations.

Fig. 14 illustrates the deployment results achieved by the three mechanisms with 50 and 90 heterogeneous directional sensors under random and corner initial configurations respectively. From this figure, we observe that the coverage performance of DVSA is limited, especially under the corner initial configuration. The reason attributes to only local info is available for DVSA to construct Voronoi polygons/cells, thus causing most sensors in densely-populated regions

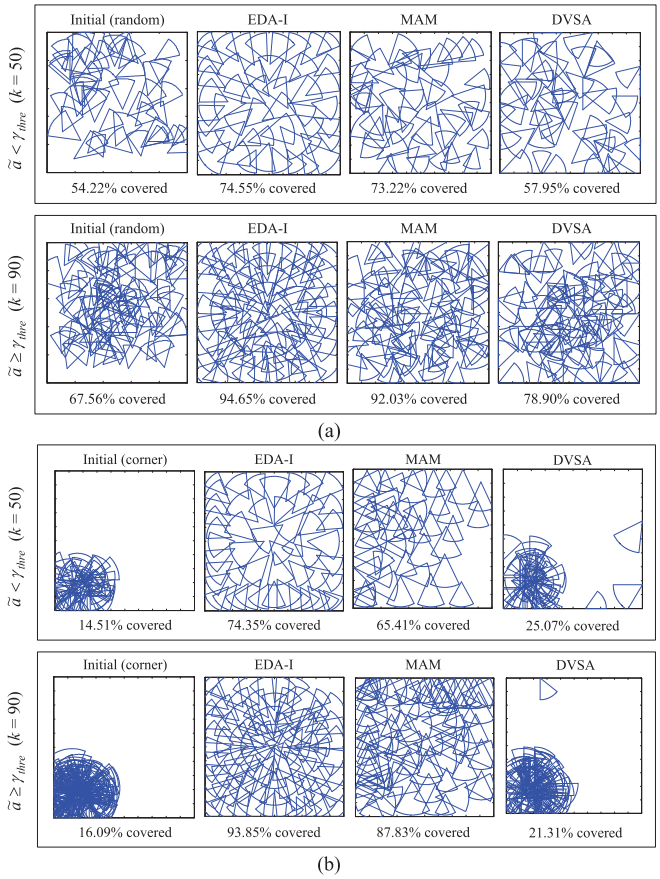


Fig. 14. Sensors distribution status after applying EDA-I, MAM, and DVSA deployment strategies, respectively, in a 300×300 area with HRA settings for 50 (upper) and 90 (lower) sensor nodes under (a) random initial configuration and (b) corner initial configuration.

unable to move outside their own polygons/cells. Consequently, we figure that Voronoi-based approaches (like our EDA-II) do require sufficient global info to construct an adequate Voronoi diagram for distributing sensors. For the MAM approach, the obtained coverage ratios are higher than DVSA but lower than EDA-I under both initial configurations. Though MAM reaches comparatively good coverage performance as our EDA-I under the random initial configuration, the energy consumed by MAM is significantly higher than EDA-I does (as to be observed in our following set of experiments). Since MAM performs rotations before movements, causing most sensors to have either vertical or horizontal working directions (sensing angles) under the corner initial configuration even after running the window-scanning process for movements, the coverage performance of MAM reaches noticeably lower ratios than EDA-I in Fig. 14b. Therefore, we infer that, to effectively improve the coverage ratio, sensors rotating actions (sensing angle adjustments) should be invoked after proper movements have been performed as our EDA-I mechanism does. Our EDA-I reaches the highest coverage ratio under both initial configurations. Interestingly, the distributed version of EDA-I achieves comparably good coverage performance to (even slightly better than) its centralized counterpart (74.55 versus 74.46 percent for $k = 50$ and 94.65 versus 94.49 percent for $k = 90$ under the same random initial configuration). This phenomenon implies that

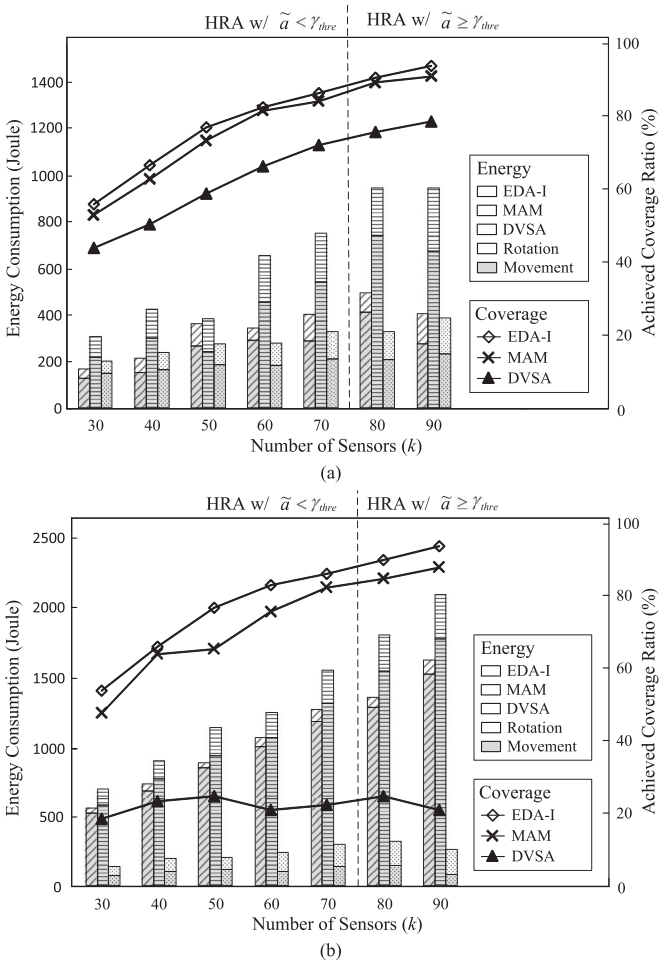


Fig. 15. Coverage ratios attained and energy consumed by EDA-I, MAM, and DVSA mechanisms under various numbers of sensor nodes in a monitored 300×300 area with (a) random initial configuration and (b) corner initial configuration.

locations info from far-away sensors (beyond 2-hop distances) is not absolutely necessary for computing proper virtual forces in our EDA-I operations. With only critical locations from the neighborhood, the virtual forces calculations employed by EDA-I are able to disregard unnecessary attractive forces imposed by far-away sensors, leading to even better coverage performance.

Next, we vary the sensors populations (densities) ranging from 30 (sparse) to 90 (dense) to observe the achieved coverage ratio and consumed energy under three deployment strategies, as shown in Fig. 15. Estimated energy includes the physical movements, rotations, and communication energy[19] consumed throughout the deployment process. This figure indicates that the DVSA approach consumes the least energy but achieves the lowest coverage ratios in all cases, due to its inability of computing a proper Voronoi diagram with only local info available. On the other hand, MAM reaches comparably high coverage ratios to our EDA-I under the random

19. Here we adopt the moving and rotating energy models from Section 6.1. For the communication energy, we follow the energy model of the communication board Crossbow MICAz (as documented in [6]) used in our prototype. From our experiments, the consumed communication energy (aggregate tx/rcv/idle energy) is relatively insignificant compared to both moving and rotating energy, and not shown in our figures.

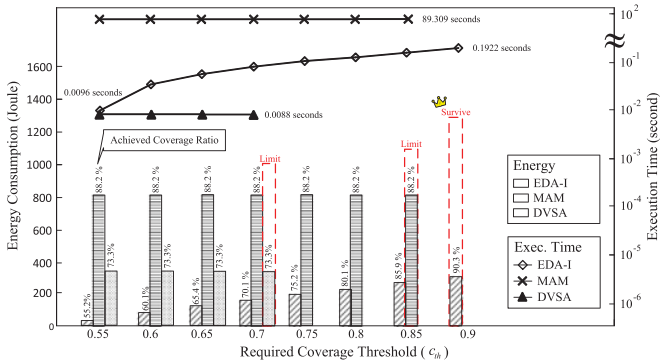


Fig. 16. Comparison of required execution time and energy consumption carried out by respective deployment strategies under certain required coverage thresholds in a 300×300 area ($k = 80$ with HRA settings) for a random initial configuration.

initial configuration, while consuming significantly greater amount of energy (more than twice for $k \geq 60$) than the EDA-I approach. This phenomenon reveals the energy-demanding feature of the MAM window-scanning process that often results in many sensors moving far distances. In contrast, our EDA-I enjoys high coverage ratios while consuming moderate energy resource, due to its capacity of keeping sensors from traveling long distances throughout the deployment process. As a result, the EDA-I deployment strategy proves to be both coverage-effective and energy-conserving.

Finally, Fig. 16 summarizes the aggregate execution time and consumed energy against different required coverage thresholds (c_{th}) under respective deployment strategies from a random initial configuration with 80 directional sensors. Estimated execution time considers the communication and computation time consumed throughout the deployment process. In our experiments, we set an acceptable execution time to be within 100 seconds. Since DVSA and MAM employ one-time (within one iteration) decisions on sensors locations and rotations, the consumed execution time and energy remain the same for all c_{th} values. On the other hand, our EDA-I is adaptable to required coverage thresholds. As the c_{th} requirement grows, EDA-I produces corresponding deployment results with coverage ratios just above the c_{th} requirements while consuming moderate energy under acceptable execution time. Fig. 16 displays that the DVSA approach can sustain the required coverage thresholds up to $c_{th} = 0.7$, while MAM sustains the coverage requirements up to $c_{th} = 0.85$. Moreover, the results suggest that MAM approach is not only energy-demanding (as explained in Fig. 15) but also time-consuming mainly due to its priority scheme (employed in the window-scanning process for sensors movements), which enforces sensors to perform deployment calculations one-by-one, greatly extending the aggregate execution time required by MAM. Consequently, MAM demands relatively higher amount of execution time compared to the other two approaches. In our experiments, for MAM to sustain beyond $c_{th} = 0.9$, five to six deployment iterations are needed (algorithmic details can be found in [14]), which prolongs the execution time to be over 300 seconds (beyond our acceptable 100-second time limit). Consequently, our

EDA-I is the only approach that survives $c_{th} = 0.9$ under the 100-second time constraint. The results validate that the EDA-I deployment strategy is capable of satisfying desired coverage thresholds while consuming moderate energy under a system-acceptable execution time requirement.

7 Conclusion

In this paper, we propose effective self-deployment algorithms, EDA-I and EDA-II, for heterogeneous directional mobile sensors. Our results demonstrate that by judiciously guiding sensors movements and rotations, the critical sensing coverage ratio can be enhanced without incurring significant energy and computation cost.[20]

Acknowledgments

This research was cosponsored in part by the Ministry of Science and Technology (MOST) of Taiwan under grant number 104-2221-E-009-089 and in part by the Delta-NCTU Cooperation Project on camera coverage research. Ting-Yu Lin is the corresponding author.

References

[1] Compro Technology Network Camera IP530/IP530W. [Online]. Available: http://www.comprousa.com/tw/product/ip530/ip530-specifications.html
[2] Compro Technology Network Camera IP570. [Online]. Available: http://www.comprousa.com/tw/product/ip570/ip570-specifications.html
[3] Crossbow Technology. [Online]. Available: http://www.xbow.com/
[4] LEGO MINDSTORMS NXT 9797. [Online]. Available: http://www.lego.com/en-US/default.aspx
[5] Logitech QuickCam Pro 4000. [Online]. Available: http://www.logitech.com/
[6] MICAz Mote Datasheet. [Online]. Available: http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf
[7] Theia Technologies PPF (Pixel-Per-Foot) Calculations. [Online]. Available: http://www.theiatech.com/papers/Resolution_calculation.pdf
[8] J. Ai and A. A. Abouzeid, "Coverage by directional sensors in randomly deployed wireless sensor networks," *J. Combinatorial Optimization*, vol. 11, no. 1, pp. 21–41, Feb. 2006.
[9] F. Aurenhammer, "Voronoi diagrams—A survey of a fundamental geometric data structure," *ACM Comput. Surveys*, vol. 23, no. 3, pp. 345–405, Sep. 1991.
[10] E. S. Biagioni and K. W. Bridges, "The application of remote sensor technology to assist the recovery of rare and endangered species," *Int. J. High Perform. Comput. Appl.*, vol. 16, no. 3, pp. 315–324, 2002.
[11] W. Cheng, S. Li, X. Liao, S. Changxiang, and H. Chen, "Maximal coverage scheduling in randomly deployed directional sensor networks," in *Proc. Int. Conf. Parallel Process Workshops*, Sep. 2007, p. 68.
[12] S. Fortune, "A sweepline algorithm for Voronoi diagrams," *Algorithmica*, vol. 2, pp. 153–174, 1987.
[13] M. A. Guvensan and A. G. Yavuz, "On coverage issues in directional sensor networks: A survey," *Elsevier Ad Hoc Netw.*, vol. 9, no. 7, pp. 1238–1255, Sep. 2011.
[14] M. A. Guvensan and A. G. Yavuz, "Hybrid movement strategy in self-orienting directional sensor networks," *Elsevier Ad Hoc Netw.*, vol. 11, no. 3, pp. 1075–1090, May 2013.
[15] A. Howard, M. J. Mataric, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed scalable solution to the area coverage problem," in *Proc. Int. Symp. Distrib. Auton. Robot. Syst.*, Jun. 2002, pp. 299–308.
[16] E. Jovanov, D. Raskovic, J. Price, J. Chapman, A. Moore, and A. Krishnamurthy," Patient monitoring using personal area networks of wireless intelligent sensors," *Biomed. Sci. Instrum.*, vol. 37, pp. 373–378, 2001.
[17] T.-Y. Lin, H. A. Santoso, W.-T. Liu, and H.-T. Liu, "An enhanced sensor deployment scheme for automated smart environments," in *Proc. IEEE Adv. Netw. Telecommun. Syst.*, Dec. 2013, pp. 1–6.
[18] T.-Y. Lin, H. A. Santoso, and K.-R. Wu, "Global sensor deployment and local coverage-aware recovery schemes for smart environments," *IEEE Tran. Mobile Comput. (TMC)*, vol. 14, no. 7, pp. 1382–1396, July 2015.
[19] M. Locatelli and U. Raber, "Packing equal circles in a square: A deterministic global optimization approach," *Elsevier Discrete Appl. Math.*, vol. 122, no. 1-3, pp. 139–166, Oct. 2002.
[20] J. Lu and T. Suda, "Differentiated surveillance for static and random mobile sensor networks," *IEEE Trans. Wireless Commun.*, vol. 7, no. 11, pp. 4411–4423, Nov. 2008.
[21] H. Ma and Y. Liu, "On coverage problems of directional sensor networks," in *Proc. 1st Int. Conf. Mobile Ad-Hoc Sens. Netw.*, 2005, vol. 3794, pp. 721–731.
[22] H. Ma and Y. Liu, "Some problems of directional sensor networks," *Int. J. Sens. Netw.*, vol. 2, no. 1-2, pp. 44–52, Apr. 2007.
[23] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proc. Int. Workshop Wireless Sens. Netw. Appl.*, Sep. 2002, pp. 88–97.
[24] S. A. Musman, P. E. Lehner, and C. Elsaesser, "Sensor planning for elusive targets," *Math. Comput. Modelling*, vol. 25, no. 3, pp. 103–115, Feb. 1997.
[25] C. Panditharathne, T.-Y. Lin, and K.-H. Chen, "Heterogeneous directional sensors self-deployment problem in a bounded monitoring area," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Mar. 2011, pp. 623–628.
[26] K.-H. Peng, T.-Y. Lin, and K.-H. Chen, "On seamless wireless sensor deployment and system implementation," *Int. J. Adv. Inf. Technol.*, vol. 3, no. 2, pp. 72–92, Dec. 2009.
[27] T.-W. Sung and C.-S. Yang, "Distributed Voronoi-based self-redeployment for coverage enhancement in a mobile directional sensor network," *Int. J. Distrib. Sens. Netw.*, vol. 2013, pp. 1–16, 2013.
[28] D. Tao, H. Ma, and L. Liu, "Coverage-enhancing algorithm for directional sensor networks," in *Proc. 2nd Int. Conf. Mobile Ad-Hoc Sens. Netw.*, Nov. 2006, vol. 4325, pp. 256–267.
[29] D. Tao, S. Tang, H. Zhang, X. Mao, X.-Y. Li, and H. Ma, "Strong barrier coverage detection and mending algorithm for directional sensor networks," *Ad Hoc Sens. Wireless Netw.*, vol. 18, no. 12, pp. 17–33, Apr. 2013.
[30] D. Tao, S. Tang, H. Zhang, X. Mao, and H. Ma, "Strong barrier coverage in directional sensor networks," *Comput. Commun.*, vol. 35, no. 8, pp. 895–905, May 2012.
[31] G. Wang, G. Cao, and T. F. La Porta, "Movement-assisted sensor deployment," *IEEE Trans. Mobile Comput.*, vol. 5, no. 6, pp. 640–652, Jun. 2006.
[32] J. Wu and S. Yang, "SMART: A scan-based movement-assisted sensor deployment method in wireless sensor networks," in *Proc. IEEE INFOCOM*, Mar. 2005, pp. 2313–2324.
[33] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," *Ad Hoc Sens. Wireless Netw.*, vol. 1, no. 1-2, pp. 89–124, Mar. 2005.
[34] D. Zhao, H. Ma, and L. Liu, "Analysis for heterogeneous coverage problem in multimedia sensor networks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2011, pp. 1–5.
[35] D. Zhao, H. Ma, and L. Liu, "Energy-efficient *k*-class coverage for collaborative classification in wireless audio sensor networks," in *Proc. IEEE Int. Symp. World Wireless Mobile Multimedia Netw.*, Jun. 2011, pp. 1–9.
[36] J. Zhao and J.-C. Zeng, "A virtual potential field based coverage algorithm for directional networks," in *Proc. Chinese Control Decision Conf.*, 2009, pp. 4590–4595.
[37] J. Zhao and J.-C. Zeng, "A virtual centripetal force-based coverage-enhancing algorithm for wireless multimedia sensor networks," *IEEE Sensors J.*, vol. 10, no. 8, pp. 1328–1334, Aug. 2010.
[38] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization based on virtual forces," in *Proc. IEEE INFOCOM*, Apr. 2003, pp. 1293–1303.

20. We plan to implement the proposed self-deployment algorithms in an operational testbed composed of mobile directional sensors with computation and communication capabilities to verify the protocol feasibility. Related prototyping experiences will be reported in an independent paper in the future.

Ting-Yu Lin received the PhD degree in computer science and information engineering from National Chiao Tung University, Taiwan. When she graduated, she received the Phi Tau Phi Scholastic Honor award. From June 2003 to February 2004, she was affiliated with the Massachusetts Institute of Technology as a research scientist. She worked for the Industrial Technology Research Institute of Taiwan from March 2004 to August 2005 as a software engineer. Afterwards, she joined the University of Illinois at Urbana-Champaign as a postdoctoral research associate in the period of September 2005 to August 2006 under both the government and university sponsorship. She is currently an associate professor with the Department of Electrical and Computer Engineering, Institute of Communications Engineering, National Chiao Tung University of Taiwan. Her research interests include wireless communications, mobile computing, WLANs/WPANs, wireless sensor/mesh networking, and green computing. She is a member of the ACM and the IEEE.

Hendro Agus Santoso received the bachelor and master's degrees in electrical engineering from the Institute Teknologi Bandung, Indonesia. He is currently working toward the PhD degree in the EECS International Graduate Program, National Chiao Tung University, Taiwan. From August 2009 to July 2012, he was a research assistant for the Microelectronics Center and ICT Research Center in ITB. His research interests include embedded systems, mobile computing, and wireless sensor networking.

Kun-Ru Wu received the BS degree in electrical engineering from the National Chung Hsing University, Taiwan, and the PhD degree from the Institute of Communications Engineering at National Chiao Tung University of Taiwan, in September 2015. He is currently a postdoctoral research associate with the Department of Computer Science, National Chiao Tung University of Taiwan. His research interests include wireless mesh networks and linux-based system prototyping. He is a student member of the IEEE.

Gui-Liu Wang received the BS degree in electrical and computer engineering from National Chiao Tung University, Hsinchu, Taiwan, in June 2013 and the MS degree from the Institute of Electrical and Computer Engineering at the same university, in September 2015. His research interests include wireless sensor networks and embedded systems.