Exploring In-Situ Sensing Irregularity in Wireless Sensor Networks

Joengmin Hwang, Tian He, *Member, IEEE,* and Yongdae Kim, *Member, IEEE*

Abstract—The circular sensing model has been widely used to estimate performance of sensing applications in existing analyses and simulations. While this model provides valuable high-level guidelines, the quantitative results obtained may not reflect the true performance of these applications, due to the sensing irregularity introduced by existence of obstacles in real deployment areas and insufficient hardware calibration. In this project, we design and implement two Sensing Area Modeling (SAM) techniques useful in the real world. They complement each other in the design space. Physical Sensing Area Modeling (P-SAM) provides accurate *physical sensing area* for individual nodes using controlled or monitored events, while Virtual Sensing Area Modeling (V-SAM) provides continuous *sensing similarity* between nodes using natural events in an environment. With these two models, we pioneer an investigation of the impact of sensing irregularity on application performance, such as coverage scheduling. We evaluate SAM extensively in real-world settings, using testbeds consisting of 14 XSM motes. To study the performance at scale, we also provide an extensive 1,400-node simulation. Evaluation results reveal several serious issues concerning circular models, and demonstrate significant improvements in several applications when SAM is used instead.

Index Terms—Model, Irregularity, Event, Sensing, Coverage, Scheduling, Similarity

1 Introduction

Wireless sensor networks are envisioned to support variety of applications such as military surveillance [3], [23], habitat monitoring [5], [29], infrastructure protection [33] and scientific exploration [31]. As a bridge to the physical world, sensing is an indispensable element of many sensor network systems. Compared to the diversified solutions produced for communication among sensor nodes, research on sensing coverage still has considerable room for improvement. One well-known but largely ignored issue is sensing irregularity. It has been known for years that sensing patterns are not regular [7], [11], [12], [20], but researchers still continue to develop, simulate, and analyze sensor network protocols that utilize a simplified theoretical sensing coverage model [1], [4], [13], [16], [21], [28], [30], [34], in which the sensing boundary is represented by a circle (a sphere in 3D) centered by a sensor. We acknowledge that the results based on this simplifying assumption could reveal high-level insights, but that such assumptions often lead to the all-too-common problem that solutions developed by simulation and analysis do not work as expected in the real world. Our work is motivated by the fact that it is difficult to accurately characterize in-situ sensing areas with theoretical models. For example, environmental impacts (e.g., obstacles) can severely affect sensing characteristics, causing irregular and non-uniform sensing patterns at different sensor nodes. Since irregularity is a common issue in sensor networks, it is unwise for developers to continue to ignore this reality. Our answer to this issue is a sensing area modeling technique called SAM, which consists of two complementary methods for sensor area modeling.

- The first method, Physical Sensing Area Modeling (P-SAM for short), features a novel way to use training events in a controlled manner. The main objective of P-SAM is to identify accurate non-parametric sensing patterns (areas), that are close to the on-the-ground truth. This is achieved by capturing the time-space relationships of controlled or monitored events and matching event positions with event detection results of individual sensor nodes. The resulting sensing area can be used to optimize the performance of many applications such as sensing coverage and event tracking.
- The second method, Virtual Sensing Area Modeling (V-SAM for short), features a lightweight way to model sensing relationship among sensors, using only observations of natural events in the environment. The main idea of V-SAM is to construct and evolve over time a series of *similarity graphs* among sensor nodes. These similarity graphs represent virtual sensing relationship among sensor nodes, which can be used to improve the application performance.

The main objective of this work is to develop two complementary in-situ modeling technologies for application designers to choose from. One can choose P-SAM to obtain sensing areas for applications that demand high-fidelity. A key challenge of P-SAM is to reconcile the conflict between the in-situ modeling accuracy and the related training cost. On the other hand, one can choose V-SAM for applications that require continuously remodeling with very low cost. The key challenge of V-

- *J. Hwang, T. He and Y. Kim are with the Department of Computer Science and Engineering, University of Minnesota, Twin Cities. E-mail: {jhwang,tianhe,kyd}@cs.umn.edu*

This work was supported by University of Minnesota McKnight-Land Grant Professorship award, and NSF grant CNS-0626614, CNS-0615063 and CNS-0626609.

SAM is how to efficiently utilize the limited information available. In summary, our contributions in this work lie in the following:

- **Measurement:** We investigate the realistic sensing patterns in existing embedded devices under various environmental settings, accessing the discrepancy between theoretical assumptions and in-situ measurements, revealing interesting observations.
- **Modeling and Validation:** We design and implement two event-driven sensing area modeling techniques. In P-SAM, we can obtain the shape of a real sensing area. In V-SAM, based on observation similarity between nodes, we develop efficient coverage scheduling algorithms to achieve desired sensing quality under realistic settings. The performance of V-SAM is examined using accurate information obtained by P-SAM. We validate the accuracy of our modeling approaches with 14 XSM motes, and an extensive simulation with 1,400 nodes.
- **Impact Analysis:** Our results serve two research purposes. First, SAM can be used to enhance the accuracy of simulation, evaluating protocols in more realistic settings. Second, SAM bridges the gap between theory and practice, integrating logical analysis with physical inputs. To our knowledge, this work is the first to study the impact of sensing irregularity on a set of protocols, including area coverage and point coverage. In these studies, we identify several serious issues with the circular model, and show improvements when SAM is used instead.

The rest of this paper is organized as follows. Section 2 describes the motivation behind our work from application perspectives. We propose P-SAM and V-SAM in Sections 3 and 4, respectively. Section 5 describes system evaluations, and Section 6 concludes the paper.

2 Related Works

Several sensing area models are used to characterize the sensing areas of individual nodes. One of the most commonly used models is *0/1 disk model*, which regards a sensing area as a disk with a certain radius centered on a sensor node. A sensor detects an event if it occurs within the disk, and it does not detect an event if it occurs outside of the disk. An enhanced disk model [2], [22], [25], [26] is based on the assumption that an event is more likely to be detected as it is closer to the sensor node. Due to the simplicity of these models, they are widely used for theoretical analysis and algorithm design. For example, many coverage scheduling algorithms [1], [4], [13], [16], [21], [28], [30], [34] are based on 0/1 disk model.

The common feature of these earlier works is to rely on a theoretical model to estimate sensing quality in ideal environments and develop applications to meet the required sensing quality. The assumption and its discrepancy from the real environment are largely specified as two parts. First, they do not consider such elements in the realistic environment as obstacles. Second, they often assume that they can obtain the key parameters required for the model, i.e., a disk size of coverage. Several projects [6], [10], [32] tried to calibrate real sensing patterns to the standardized units. For example, sensor array calibration based on constant target tracking was proposed in [6]. The concept of macro-calibration for localization was introduced in [32]. Auto-calibration for acoustic sensor network was designed and implemented in [10]. Overall, the objective of calibration is to obtain mapping parameters to represent real world. However, calibration in large-scale sensor network still has lots of issues. In addition, it does not provide a general solution to the performance degradation caused by obstacles, an important factor in sensing irregularity.

Having observed the limitations of these simplifying models, several pioneering projects have been proposed to design algorithms and protocols [18], [19], [27] without any prior assumptions on the sensing coverage. Koushanfar et al. [18] proposed an energy efficient sleeping coordination for environmental monitoring (temperature sensor, humidity sensor, etc). Using the correlation between sensor nodes, a model is constructed to predict the values of some sensors from the values of other sensors. Krause et al.[19] dealt with sensor placement problems assuming no knowledge about sensing pattern. Based on the data of sensing values, their algorithm selects near-optimal locations of sensor nodes so that the number of sensor nodes and the cost are reduced. While both works show their effectiveness in dealing with sensing irregularity, they are specific solutions on a case-by-case basis. Instead, the objective of our work is to provide a generic solution for sensing irregularity that is directly comparable to the widely used circular 0/1 sensing model in the literature [1], [4], [13], [16], [21], [28], [30], [34]. This work is an extended version of our earlier paper [17]. In this extended version, we provide new sensing coverage abstraction method, analyze P-SAM overhead, and add another application related to sensing coverage.

3 Physical Sensing Area Modeling (P-SAM)

In this section, we introduce the design of P-SAM. We focus on static sensor networks (i.e., with no mobility), which are the case for most existing deployed sensor systems [29], [31]. We first describe our design and analysis as conceptually independent of the type of events used. Later on, we use Passive InfraRed (PIR) motion sensors as specific examples in P-SAM implementation.

3.1 Main Idea

The main idea of physical sensing area modeling is to relate the location of events to the event detection results of individual sensors. Events can be intentionally generated in the space where the sensor nodes are deployed, or we can monitor natural events and collect

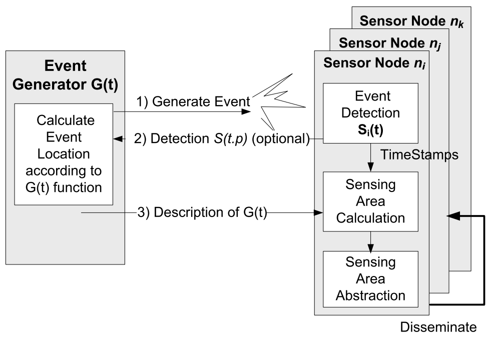


Fig. 1. P-SAM Architecture

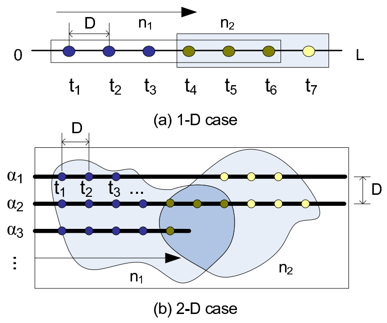


Fig. 2. Regular Training

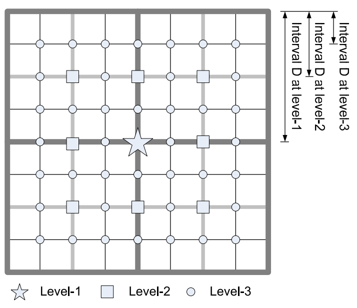


Fig. 3. Hierarchical Partition

```
Algorithm 1 Regular $G(t)$ Process
Output: $P_i$: The sensing area of $n_i$.
1: $T = \emptyset$ //an empty set of timestamps
2: repeat
3:     An event $e(t,p)$ is created at time $t$ and location $p(x,y)$ according to $G(t)$
4:     if node $n_i$ detects event $e(t,p)$, i.e. $S_i(t,p) = 1$ then
5:         it stores the timestamp $t$ into set $T$
6:     end if
7: until $G$ stops generating events
8: Event generator $G$ disseminates the description of $G(t)$ to all nodes
9: Node $n_i$ obtains a set of locations $P_i$ by correlating $G(t)$ with $T_i = \{t_1^i, t_2^i, \dots, t_n^i\}$
10: $P_i$ is a set of positions $p$ where $S_i(t,p) = 1$
```

information on their locations. We call both controlled and monitored events *training events*. An event could be, for example, the presence of an object in an area or a light spot projected on a set of sensors. The obtained sensing area can be input to an existing coverage scheduling algorithm [34] to improve sensing quality.

Formally, an event can be defined as a detectable phenomenon $e(t,p)$ that occurs at time t and at location $p \in A \subset \mathbb{R}^k$ $(k = 1,2,3)$. Without loss of generality, we use $k = 2$ (2-dimensional plane) in the rest of the paper. To identify sensing area, we need to match a relationship between the time t and location p. In other words, a set of training events can be described as the event locations over the discrete time: $G : \mathbb{R} \to \mathbb{R}^2$, where $G(t) = p_t = (x_t, y_t)$ and $t \in \{t_1, t_2, ..., t_n\}$. In case of continuous events, discrete events can be obtained by sampling a continuous event with a certain interval.

Fig. 1 shows the system architecture of P-SAM, which consists of two major parts: an event generator G and a set of sensor nodes $n_i (i \in \mathbb{N})$. The event generator G is a function to assign a physical point to a discrete time according to which a sequence of events $e(t,p)$ are generated, (Step 1 in Fig. 1). We define $S_i(t,p)$ as the detection function of node n_i, if node n_i can detect event $e(t,p)$, $S_i(t,p) = 1$; otherwise $S_i(t,p) = 0$. In case of detection, sensor nodes store the timestamp t locally. By the end of training, G can either collect the time-stamps from sensors (Step 2) or disseminate the description of $G(t)$ to whole network (Step 3). By inputting the time

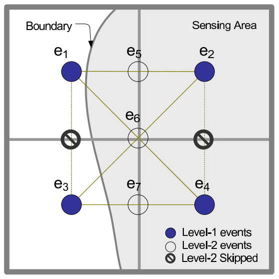


Fig. 4. Level of Details

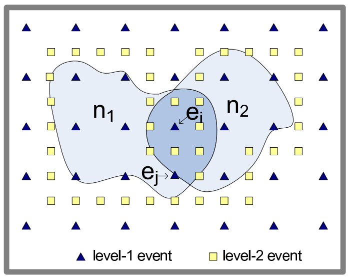


Fig. 5. Hierarchical Training

stamps into $G(t)$, a set of timestamps $T_i = \{t_1^i, t_2^i, \dots, t_n^i\}$ from node n_i can be converted to a set of locations $P_i = \{p_1^i, p_2^i, \dots, p_n^i\}$. The location set P_i can be used to directly describe the sensing area of node n_i, or it can be transformed to a polygon.

3.2 Design of Event Generator $G(t)$

Since the overhead and accuracy of the sensing modeling is largely determined by $G(t)$, it is important to consider several solutions to optimize $G(t)$ under different system configurations.

3.2.1 Regular $G(t)$

To illustrate the basic functionality of an event generator, we start with a simple sensor system in which the sensing area of a node is a line segment as shown in Fig. 2a. We intend to find out the portion of the line included in the sensing ranges of sensor node n_1 and n_2. To achieve this, the event generator creates discrete point events along this line $[0, L]$ with constant speed v with an *interval* D. Formally, $G(t) = t \cdot v$, where $t = kD/v$ and $0 \leq k \leq L/D$. A large k is requred for the detailed coverage shape with high accuracy (the trade-off between the accuracy and training cost is described in [17]). In the example in Fig. 2, a sensor node n_1 collects a set of six timestamps $T_1 = \{t_1, t_2, \dots, t_6\}$ at which the events are detected. Using function G, the timestamps are converted to a set of locations $P_1 = \{t_1 v, t_2 v, \dots, t_6 v\}$. The sensing coverage of sensor n_1 is defined as the line segment that covers P_1.

The regular training can be generalized to the case when the events occur in a plane. Fig. 2b shows this approach. In this case, training area A is divided into

Algorithm 2 Hierarchical $G(t)$ process

```
Output: $P_i$: The sensing area of $n_i$.
1:  $G(t)$ starts with level-1 events $e(t,p)$ (The number of level-1 events is decided by the minimum sensing area)
2:  Node $n_i$ reports $S_i(t,p)$ for all level-1 events
3:  repeat
4:      for all level-$k$ adjacent pairs $e(t_m,p_m)$ and $e(t_n,p_n)$ do
5:          if any node detects only one event && no event is generated at position $\frac{p_m+p_n}{2}$ before then
6:              Generate a level-$(k+1)$ event at position $\frac{p_m+p_n}{2}$
7:          end if
8:      end for
9:      $k = k+1$
10: until ($k$ = Maximum Level)
11: $P_i$ is a set of positions $p$ where $S_i(t,p)=1$
```

several lines $\alpha_1, \alpha_2, \ldots$, and the events are generated following the lines. In addition to the progressive scanning, $G(t)$ function of the regular training can use an arbitrary sequence of natural events as long as the position of the natural events can be acquired along with detection results $S(t,p)$. The detailed operations to identify the sensing area of a single node n_i are described in Algorithm 1.

3.2.2 *Hierarchical* $G(t)$

Hierarchical $G(t)$ is motivated by the observation that the boundary of a sensing area requires more detail than the area in the middle of coverage. With hierarchical $G(t)$, we can reduce the number of events required to obtain the same accuracy as regular $G(t)$.

As shown in Fig. 3, level-1 events divide the area into 4 sub-areas, and level-2 events divide the area into 16 sub-areas. In general, level-i events divide an area into 4^i sub-areas. *Interval D at level-i* is the distance between adjacent sub-areas' centers. If an event is a level-i event, it is also a level-j event ($j \geq i$). Two events are said to be *adjacent* (or a pair) if they are neighboring each other vertically, horizontally or diagonally (e.g., an event could have a maximum of 8 adjacent events). Two adjacent events are said to be *boundary pair* if only one of two adjacent events is within a sensing range of some node. (e.g., e_1 and e_5 in Fig. 4 form a boundary pair). The event in a boundary pair is called a *boundary event*.

The main idea of hierarchical $G(t)$ is to *recursively generate new events in the middle of boundary pairs.* It works in a way similar to the binary search within a two-dimensional space. We describe detailed operation of hierarchical $G(t)$ in Algorithm 2.

3.2.3 *A Walkthrough of Hierarchical* $G(t)$

We illustrate the main idea for finding the sensing area of one sensor using hierarchical training. Fig. 4 shows four level-1 events e_1, e_2, e_3 and e_4 that are generated coarsely at time $T = \{t_1, t_2, t_3, t_4\}$. By definition, these events are adjacent to each other. In the example, the sensing area of a node covers about half of the area; therefore, the event generator G obtains the detection results $S(t_1,p_1) = S(t_3,p_3) = 0$ and $S(t_2,p_2) = S(t_4,p_4) = 1$. According to lines 4 - 8 in Algorithm 2, we compare the value $S(t,p)$ for each pair of adjacent events. In the example, since $S(t_1,p_1) = S(t_3,p_3)$ and $S(t_2,p_2) = S(t_4,p_4)$, no event is generated in the middle of e_2 and e_4, nor in the middle of e_1 and e_3. These skipped locations are assumed to have the same value as $S(t_2,p_2) = S(t_4,p_4)$ and $S(t_1,p_1) = S(t_3,p_3)$, respectively. However, since $S(t_1,p_1) \neq S(t_2,p_2)$, $S(t_1,p_1) \neq S(t_4,p_4)$, $S(t_3,p_3) \neq S(t_4,p_4)$, we need to provide an additional level of detail by generating three new events, e_5, e_6 and e_7. These events are at the middle of selected pairs of adjacent events at times t_5, t_6, and t_7, as shown in Fig. 4.

Hierarchical $G(t)$ works recursively. After new events are added, new adjacent pairs can be created. For example, after we add e_5, e_6, and e_7, the event e_5 has new adjacent pairs $e_5 \leftrightarrow e_1$, and $e_5 \leftrightarrow e_2$, and $e_5 \leftrightarrow e_6$. Such new pairs are checked with the same procedure detailed in lines 4-8 in Algorithm 2 until we reach the maximum level of detail we defined. For a sensor n_i, all values in a set S collected at all levels of detail are used for the calculation of its sensing coverage.

Hierarchical $G(t)$ can be generalized for any number of sensors involved where a certain area can be covered by more than one sensor. We need to check whether two adjacent events, e_i and e_j, have the same value of $S(t_i,p_i)$ and $S(t_j,p_j)$ for *all* neighboring sensors. In other words, two adjacent events are said to be a *boundary pair* as long as there exists a sensor that detects only one event. Fig. 5 gives an example. The area is covered by two sensor nodes, n_1 and n_2. After level-1 event generation, the detection results of two adjacent events are compared. Although node n_1 detects both events, e_i and e_j, node n_2 detects only e_i. Therefore, e_i and e_j form a boundary pair (of n_2), and a new event should be generated in the middle of the two events.

3.3 Analysis of P-SAM Overhead

In this section, we analyze the cost for the regular training and hierarchical training. We focus on the number of events generated. The key purpose of this analysis is to identify the method that minimized total cost for each given number of nodes. Without loss of generality, we assume that the N sensor nodes are randomly deployed in area A, which is X by X meters. To facilitate comparison, we set the minimum event interval $D = \frac{X}{2^I}$, where I is the highest level for hierarchical G(t).

3.3.1 *Overhead in the Regular Training*

One of nice features of the regular training is that event overhead is independent of the number of nodes within the network. To cover area A, event generator G divide area into $(\frac{X}{D})^2$ sub-areas, so the total number of events generated, denoted as $O_{reqular}$, is:

$$O_{reqular} = (\frac{X}{D} - 1)^2 = (2^I - 1)^2. \qquad (1)$$

Suppose G can generate K events per second. It takes $\frac{(2^I-1)^2}{K}$ seconds to finish.

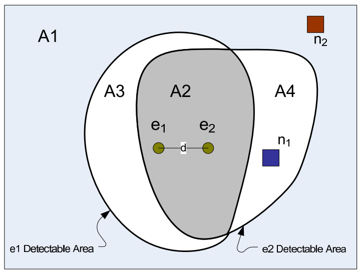


Fig. 6. Probability two adjacent events are detected by same set of sensors

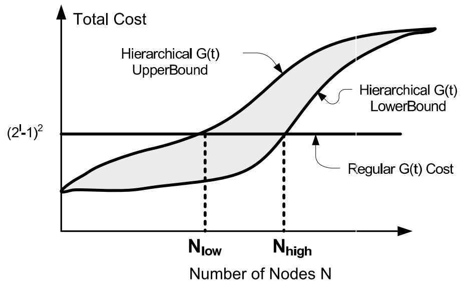


Fig. 7. Preferable regions for regular and hierarchical $G(t)$

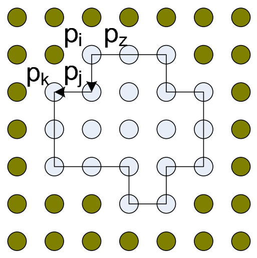


Fig. 8. Polygon abstraction

3.3.2 Overhead in the Hierarchical Training

For the hierarchical training, the number of events is closely related to the number of sensor nodes and its sensing coverage range. Before estimating the event overhead, we calculate the probability $P_{pair}(d)$ that a pair of adjacent events e_1 and e_2 with distance d is a boundary pair. By definition, if two adjacent events are detected by different sets of sensor nodes, they form a boundary pair. Fig. 6 visualizes the proof. In Fig. 6, area A is partitioned into four subareas $A1, A2, A3$ and $A4$ by two irregular circles. Obviously, if any sensor node is located within either the A3 or A4 area, the event e_1 and e_2 will be detected by different sets of nodes. Since a node is randomly deployed in area A, the probability that a node falls in to $A3$ and $A4$ is $\frac{A3+A4}{A}$. For N nodes, the probability of at least one of them falling in $A3$ and $A4$ is

$$1 - \left[1 - \frac{A3 + A4}{A}\right]^N \tag{2}$$

This probability is essentially $P_{pair}(d)$. If we assume further the detectable area of e_1 and e_2 can be approximated by two circles with sensing radius R, we can obtain the worst-case event overhead for hierarchical $G(t)$. Given an X by X area, R and d, following a simple geometric derivation, we can refine the $P_{pair}(d)$ value as:

$$P_{pair}(d) = 1 - \left[1 - \frac{2\pi R^2 - 4R^2\cos^{-1}\left(\frac{d}{2R}\right) + d\sqrt{4R^2 - d^2}}{X \cdot X}\right]^N \tag{3}$$

Given $P_{pair}(d)$, we can now calculate the probability $P_{node}(d)$ that an event e is a boundary event within the distance d. Obviously, if e forms a boundary pair with any adjacent event, it is a boundary event. Without edge effect, each event has 2 horizontal(with distance d), 2 vertical (with distance d) and 4 diagonal neighboring events (with distance $\sqrt{2}d$). Therefore, $P_{node}(d)$ is:

$$P_{node}(d) = 1 - (1 - P_{pair}(d))^4(1 - P_{pair}(\sqrt{2}d))^4 \tag{4}$$

Now we are ready to estimate the event cost in hierarchical G(t). In general, at level-i, there are $(2^i - 1)^2$ events (note one event belongs to its level and above). To ensure accuracy, all level-I boundary events (I is the highest level) must be generated. This gives a lower-bound of event cost, denoted as O_{min}:

$$O_{\min} = P_{node}(D)(2^{\mathrm{I}} - 1)^2 \tag{5}$$

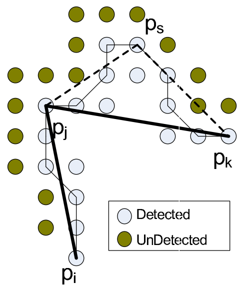


Fig. 9. Polygon simplification

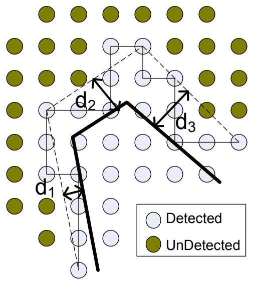


Fig. 10. Inscribed polygon

where $D = \frac{X}{2^I}$ is the minimum event interval. We note O_{min} is a tight lower-bound. When $\lim_{N\to\infty} P_{node}(d) = 1$, O_{min} equals the number of events generated in the regular training.

The upper-bound of event cost O_{max} can be obtained by adding up the numbers of boundary events at all levels. Because a level-i events is also level-j as long as $j \geq i$, we might introduce duplicated counts by summing the cost at all levels, hence O_{max} can only serve as an upper-bound. Since during the training, hierarchical $G(t)$ records the previously generated events (to eliminate duplicates), in the worst case, hierarchical $G(t)$ introduces the same event overhead as regular $G(t)$. Therefore O_{max} can be formulated as follows:

$$O_{\max} = \min\left(\sum_{i=1}^{I} P_{node}(2^{I-i}D)(2^i - 1)^2, (2^I - 1)^2\right) \tag{6}$$

We note that both $O_{\max}$ and O_{min} increase when the number of nodes N increases. Especially according Equations 3 and 4, when $N \to \infty$, $P_{node}(d) \to 1$. Therefore both $O_{\max}$ and O_{min} converge to $(2^I - 1)^2$, which is the cost of regular training. This indicates that the hierarchical training is more efficient when the node density is low.

3.3.3 Overhead Comparison of two $G(t)$

There are two factors affecting the overhead of $G(t)$: First, the overhead in terms of the number of events generated. Second, the overhead in terms of the number of $S(t,p)$ messages reported to event generator G. As for event overhead, we have proven that the hierarchical training always has a lower or equal event overhead than the regular training, especially when the node density is low. As for the message overhead, the nodes in regular training do not need to report the detection results to the generators, while the hierarchical training uses detection reports $S(t,p)$ about the level-i events to adjust the generation of the level-$(i{+}1)$ events. By taking both types of overhead into account, we can identify the preferrable region for hierarchical and regular $G(t)$.

Total overhead can be calculated by assigning different weights to the event and message overhead. Without loss of generality, Fig. 7 shows the total overhead under an equal weight. Based on Equation(1), the cost of regular $G(t)$ is a flat line at value of $(2^I - 1)^2$. The cost of hierarchical $G(t)$ lies in the gray area, enclosed by two bounds, which are calculated by Equations 5 and 6. Comparing two bounds of the hierarchical $G(t)$ with the regular $G(t)$, we can obtain two critical values N_{high} and N_{low}, respectively. When the number of nodes N is below N_{low}, hierarchical $G(t)$ is a better choice. When the number of node N is above N_{high}, the regular training $G(t)$ is the right choice. When the number of the nodes falls between N_{low} and N_{high}, the total cost is affected by the distribution of sensor nodes. A high dispersion leads to more cost in the hierarchical training. The worst case for hierarchical $G(t)$ is when sensing areas of sensor nodes do not overlap.

3.4 Sensor Area Abstraction

In the basic SAM design, we use a set of locations, P_i to represent the sensing area of node n_i. Evidently, this representation based on raw sampling data requires excessive memory, especially when the sensing area is large. It also introduces unnecessary message overhead, when neighboring nodes share the sensing area information. To address this issue, we propose a set of model abstraction techniques. Essentially, the goal is to reduce the complexity of a model sufficiently, without suffering a noticeable loss in accuracy. Specifically, we abstract a set of discrete sampled locations as a polygon. In addition, we study how to further simplify the polygon when the abstracted polygon is still too large to store/transmit. We design and implement three polygon abstraction methods, in which the first method is a base for the second and third methods.

1) **Polygon Abstraction through Wrapping:** For a sensor node n_i, given a set of event locations P' (associated with $S=1$) and a set of event locations P''(associated with $S=0$) in training process, generate a polygon H that covers P' but not P''. (Fig. 8)
2) **Polygon Simplification using Douglas-Peucker (DP) Algorithm [8]:** Given a polygon H with n vertices, find a polygon H' with n' vertices, where the shape of H' is close to H and $n' < n$. DP algorithm generates an accurate approximation in $O(n^2)$ [8], $O(n \log n)$ [15], where n is the number of vertices. Besides the DP algorithm, the vertex reduction algorithm is a fast $O(n)$ algorithm where vertices close to each other are reduced to a single vertex. (Fig. 9)
3) **Inscribed Polygon Simplification:** Given a polygon H with n vertices, find an inscribed polygon H' with n' vertices, where $n' < n$ and subtracted area of H' from H is minimized. (Fig. 10)

Algorithm 3 P-SAM Coverage scheduling [34] for event detection implemented on node n_i

```
Input: a set of locations of interest, $\{l_1^i, l_2^i, \ldots, l_m^i\}$ covered by node $n_i$
1: Exchange information $\{l_1^i, l_2^i, \ldots, l_m^i\}$ with neighbors.
2: Select a random time $R_i$ and exchange with neighbors.
3: $t_{start} \leftarrow R_i$, $t_{end} \leftarrow R_i$
4: for each $l_k^i$, $k = 1, \ldots, m$ do
5:     Find every neighbor covering $l_k^i$, and sort $R_i$ and every neighbor's random time in increasing order.
6:     $[t_{start}, t_{end}] \leftarrow [t_{start}, t_{end}] \cup [\frac{R_i + Pred(R_i)}{2}, \frac{Succ(R_i) + R_i}{2}]$
7: end for
8: Schedule node $n_i$ to wake up at $t_{start}$ and sleep at $t_{end}$.
```

3.5 Application: P-SAM Guided Coverage

We can use the output of P-SAM to improve the performance of many sensing-driven applications. As a specific example in this work, we apply P-SAM to the coverage scheduling algorithm proposed by [34] to show its effectiveness.

Algorithm 3 describes how the coverage scheduling algorithm in [34] can be built on top of P-SAM. The sensing phase is divided into rounds with equal duration. Within each round, each node needs to decide when to sleep and when to work (in order to save/balance energy). To do that, each node n_i keeps its sensing area as a set of locations (points) it covers, $\{l_1^i, l_2^i, \ldots, l_m^i\}$ [Line 1 of Algorithm 3]. It selects a random time R_i in range of round starting time and ending time, and disseminates it to its neighboring nodes [Line 2]. For each location l_k^i in the location list, it finds its neighbors that cover the location. Let $Pred(R_i)$ be the largest random time of neighbors smaller than R_i. The node n_i's wake-up time is the middle of $Pred(R_i)$ and R_i. Similarly, $Succ(R_i)$ is the smallest random time of neighbors larger than R_i. Then, the node n_i's sleep time is the middle of R_i and $Succ(R_i)$. For each location l_k^i, node n_i's wake-up and sleep time is determined [Line 4-5] in this way. The minimum wake-up time over all locations is chosen as the final wake-up time, and the maximum sleep time over all locations is chosen as final sleep time [Line 6].

In the circular model, the sensing area of a sensor node is a circle with a certain radius centered at the sensor node. Thus, all physical points contained within a circle are provided as an input to Algorithm 3. If we use P-SAM, we regard a sensing area as a collection of the locations obtained during training process. In this case, the collected set of locations is provided as an input.

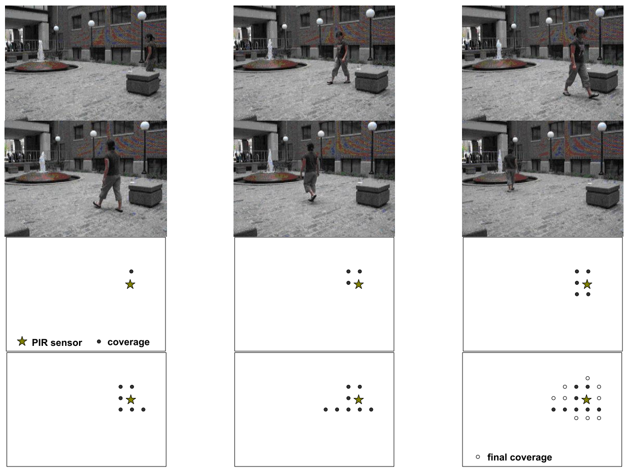


Fig. 11. Mapping Event Detection to the Event Position Using Image Capture at Time $t = 6, 8, 10, 16, 17, 20$ (sec); last map Includes Additional Training Results by $t = 40$ (sec)

3.6 Implementation of P-SAM

We use ExScal XSM motes [9] to obtain empirical results on irregular sensing patterns. Four PIR sensors, each with $90°$ view, are attached to a XSM mote to provide a full $360°$ view of sensing. PIR sensors detect movements through changes in infrared radiation, which is caused by walking persons or moving vehicles. The sensing area would change slowly over time due to the changes in ambient conditions and the energy condition of the nodes. However, from our experience, we find that the PIR sensing area is relatively stable; there is no significant difference unless the environmental factors change significantly. Thus, several trainings over a large time interval would be enough. For example, we measure the sensing area once during day and once at night, and we also measure the sensing area in the winter and in the summer. We trade off the model accuracy over time with the cost to refresh the model.

We adopted the regular training approach, but instead of training the motes using parallel lines as in Fig. 2b, we used monitored events (i.e., natural movements of a person). To map the event time to the event position, we used a camcorder during training. Then the event detection time is compared to the camcorder capture time and converted to the location included in the sensing area. For example, in Fig. 11 the camcorder captures the positions of a person at time $t = 6, 8, 10, 17, 18, 20$ (sec), converts the detection time of the PIR sensor to the corresponding position of the trainer in the picture, and projects the position into the plan.

4 Virtual Sensing Area Modeling (V-SAM)

Clearly, the strength of P-SAM is in its high accuracy in sensing modeling. It is achieved, however, at the cost of controlled training. While P-SAM is useful in scenarios where sensing accuracy is highly desired, we need a complementary solution that is suitable for scenarios where cost is the paramount concern and the sensing area evolves relatively quickly over time. In this section, we propose the lightweight design of V-SAM, which requires no controlled events. The V-SAM modeling technique is especially useful when the events occur frequently, and when we want to capture the coverage without micro-control in areas with unknown obstacles.

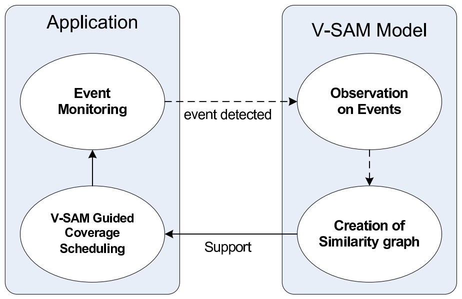


Fig. 12. V-SAM and Coverage Scheduling Built-Upon It

4.1 Main Idea

Fig. 12 shows the process of V-SAM and how applications can be built upon it. We assume if two nodes are neighbors in sensing range they are neighbors in communication range. Each sensor node exchanges sensing values for detected events and calculates similarity between neighboring nodes. The resulting similarity graph represents virtual sensing relations among the sensor nodes. On top of V-SAM, applications can be built. For example, in sensing coverage, nodes can coordinate their working schedule based on the similarity graph. The highlight of V-SAM is the continuity of the V-SAM modeling process, i.e., the similarity graph can be continuously updated/refreshed with upcoming events in the system.

4.2 Design of V-SAM

V-SAM consists of two main procedures: similarity measure and similarity graph construction.

4.2.1 Measuring Similarity

In V-SAM, an event is defined as a detectable phenomenon that occurs at time t at location p, which is unknown *a priori*. As shown in Fig. 13, nodes are roughly synchronized with each other [24], and time is divided into equal *round* with duration T_{update}, which is a parameter to control how often the sensing model is refreshed. Each round is further divided into m equal duration intervals, each of length T_{span}.

For each round, each node n_i stores its observation vector $\{o_1^i, o_2^i, \ldots, o_m^i\}$ obtained through discrete sampling at $T_i = \{t_1^i, t_2^i, \ldots, t_m^i\}$. After collecting the event observations, at the end of round each node exchanges the observation vector, which is used to calculate similarity between nodes. Specifically, we use $\mathcal{P}$-norm to measure the similarity between two observation vectors by node n_i and node n_j as follows:

$$d(i, j) = \sqrt[\mathcal{P}]{\sum_{k=1}^{m} |o_k^i - o_k^j|^{\mathcal{P}}} \quad (7)$$

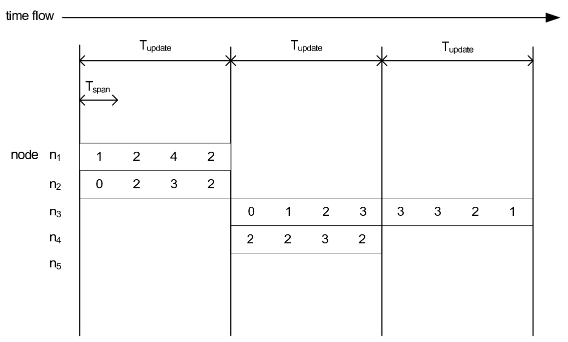


Fig. 13. Time Series of Event Observation

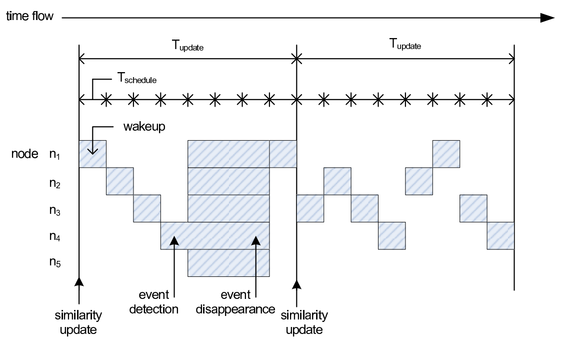


Fig. 14. V-SAM Guided Coverage Scheduling

where $\mathcal{P}$ can be $1, 2, \ldots, \infty$.

Let o_k^i, o_k^j be in a range of $[o_{min}, o_{max}]$. Then, $d(i,j)$ have minimum value $d_{min} = 0$ when two motes have the same observations, and maximum value $d_{max} = \sqrt[\mathcal{P}]{m \times (o_{max} - o_{min})^{\mathcal{P}}}$ when they have completely different observations. To make the range d_{min} to d_{max} normalized to 1 to -1, we map $d(i,j)$ to $((-1) - 1) \times \frac{d(i,j) - d_{min}}{d_{max} - d_{min}} + 1$. This is the difference of a new range times the proportion of $d(i,j)$ in the old range plus the starting value of the new range. *The resulting value is closer to* 1 *if two motes have similar observations, while it is closer to* (-1) *if they have different observations*.

To estimate the similarity over time, we use an exponential moving average method. The average similarity in the n^{th} round, $\hat{d^n}(i,j)$ is updated differently, depending on whether there is any event detection in the round. When a node detects an event, new $d^n(i,j)$ is used to update $\hat{d^n}(i,j)$. Otherwise, *we use an aging factor* β *to gradually attenuate the similarity among nodes to 0*. The rationale behind the aging factor is to forget the similarity observed a long time ago, which cannot accurately reflect the current situation. The average similarity over n rounds ($n \geq 1$) is calculated by:

$$\hat{d^n}(i,j) = \begin{cases} \text{if event is detected} \\ \quad \alpha \times \hat{d}^{n-1}(i,j) + (1-\alpha) \times d^n(i,j) \\ \text{otherwise} \\ \quad \beta \times \hat{d}^{n-1}(i,j) \end{cases} \tag{8}$$

We provide an example in Fig. 13. When we compute the similarities between nodes n_1 and n_2, we make the observation vector of node n_1 as $\{1, 2, 4, 2\}$ and observation vector of node n_2 as $\{0, 2, 3, 2\}$. Similarly, two observation vectors of nodes n_3 and n_4 can be made. *Initially, the default similarity value of* $d^0(i,j)$ *is left to* 0.

4.2.2 Building Similarity Graph

Now that we have measured similarities, we want to represent the *sensing relations in a graphical method in a given time space*. We use a graph $G(V, E(t))$ to represent a set of sensor nodes and similarities among themselves at a certain *time slot* t (a duration of each time slot t is set to T_{sch} in Section 4.3, which will be explained in the section). The set V is a complete set of N sensor nodes in the network, and $E(t)$ is a set of edges among nodes. A graph is not static, and changes over time. For each time slot t, an edge between nodes n_i and n_j is added with probability proportional to the degree of similarity. More specifically, after n^{th} round, at a certain time t, an edge $e(i,j)$ belongs to $E(t)$ if and only if Equations (9)-(10) are satisfied.

$$R^t(i,j) \leftarrow \text{Rnd}(i \mid\mid j \mid\mid t) \tag{9}$$

$$R^t(i,j) \; < \; w \cdot \hat{d^n}(i,j) \tag{10}$$

where $i < j$, Rnd(s) is a random number generated *in range* -1 *and* 1 using s as a seed, || is a concatenation operation, and term w represents the weight applied to each similarity. We concatenate i and j in increasing order to make the random number generated in nodes n_i and n_j the same. Then, an edge is added with probability:

$$\Pr[\ R^t(i,j) \; \leq \; w \cdot \hat{d^n}(i,j)\ ] \tag{11}$$

Over a set of time slots, for similarity $\hat{d^n}(i,j)$ with a small negative or positive value (i.e.,near zero), two nodes n_i and n_j become neighbors more randomly. On the other hand, for $\hat{d^n}(i,j)$ with big negative value (i.e. near -1), two nodes are less likely to be neighbors, and for $\hat{d^n}(i,j)$ with large positive value (i.e., near 1), two nodes are more likely to be neighbors. The rationale behind this design is that neighbors with similar view are connected more frequently, while neighbors with dissimilar view are disconnected more frequently. The neighbors determined neither similar nor dissimilar are randomly connected.

4.3 Application: V-SAM Guided Coverage

Most existing coverage scheduling algorithms purely depend on the assumption that the deployment area is open space and the sensing area of individual sensor is uniform (circular). Obviously, this assumption does not hold well in the real world. Differently, our proposed coverage scheduling algorithm does not require such assumption. Instead we schedule nodes' sleep and wakeup time, based on in-situ similarity graph calculation. This is done by turning off nodes with similar sensing experience at different time slots to save energy consumption, and by turning on nodes with dissimilar sensing experience to work together to achieve required sensing quality.

Algorithm 4 Coverage scheduling for event detection implemented on node n_i

```
Input: New observations $o_i$ and $o_j$ for every physical neighbor $n_j$ (neighbor within communication range).
 1: for each round $n$ do
 2:     if At the beginning of a round then
 3:         Compute the similarity $\hat{d}^n(i,j)$ with every physical neighbor $n_j$.
 4:         for each scheduling slots $t$ within current round do
 5:             if $p_i^t > p_j^t$, for every node $n_j$ such that $R^t(i,j) < w \cdot \hat{d}^n(i,j)$ then
 6:                 Assign the node $n_i$ to wake up at time slot $t$.
 7:             end if
 8:         end for
 9:     else
10:         Node $n_i$ turns on and off according to the schedule calculated at the beginning of the round.
11:         If a new event is detected, node $n_i$ inform it of all physical neighbors at their wake-up slot, and they record/report new observations.
12:     end if
13: end for
```

4.3.1 Basic Algorithm

In our work, we propose a distributed heuristic algorithm as detailed in Algorithm 4. The key idea is that nodes with similar observations are restrained from waking up simultaneously, as the additional information does not increase significantly. Specifically, our algorithm works as follows:

As shown in Fig. 14, the time is divided into equal length *round* with duration T_{update}. Each round is further divided into multiple *time slots* with duration T_{sch}. At the beginning of each round, similarity is calculated [Line 1-3 in Algorithm 4] using the observation obtained in the previous round. The similarity graph changes for each time slot t. For each basic time slot t, a node decides whether to add an edge and make a physical neighbor (neighbor within communication range) as a neighbor on similarity graph following Equations (9)-(10) [Line 4-8]. For each node n_i, we define the priority of node i at time t as

$$p_i^t = \text{Rnd}(i \mid\mid t) \mid\mid i. \tag{12}$$

where Rnd(s) is a random number selected from elements of $\{1, 2, \ldots, N\}$, and N is a natural number greater than the number of deployed nodes. We concatenate unique node ID i, in order to ensure that no two nodes have same priority at time slot t. For each basic time slot t, a node calculates locally its own priority as well as the priority of its neighboring nodes (of similarity graph). Since all nodes share the same random number generator, the computation of priority requires no communication among nodes. If the node's own priority is higher than all its neighbors, this node is scheduled to be activated in the time slot t. After nodes decide their working schedule at the beginning of the round, they simply follow the working schedule in the rest of the round [Line 10].

New observations can be obtained if events are detected by one of active nodes [Line11]. Event detection is informed to other nodes in their wakeup time slot, and they work together to monitor the detected event. The observations afterward are collected at a sink, and observation message is overheard by each physical neighbor. If no event is detected during T_{update} period, the similarity graph decays with the aging factor β.

As a case study, Fig. 15 illustrates the graph representation of the similarities among sensor nodes and coverage scheduling based on the similarities. Nodes with similar observations are connected to each other via an edge, which results in two clusters in two sides and one sensor node crossing over the clusters. The priority of sensor node n_1 is $p_1 = 17$, which is highest among any other sensor nodes connected via an edge. Thus, the sensor node n_1 is awake at the time slot. In addition, nodes n_5, n_7, n_{10} are selected to be awake at the time slot by winning the highest priority.

With the immature training, the performance of V-SAM guided coverage scheduling should not be worse than random coverage scheduling. This is achieved by constructing similarity graph in probabilistic sense according to the degree of similarity as in Section 4.2.2.

4.4 Implementation of V-SAM

We have implemented and evaluated the V-SAM system at the ground floor of one of our university libraries, as shown in Fig. 16. The area was selected because it reflects a realistic environment, full of bookshelves, tables, small rooms and other obstacles. A monitoring system in such an environment is useful, such as to automatically turn on the lights in the bookshelves area when motion is detected. In the study area, we are interested in monitoring behaviors of students, especially disturbing movements, in the library. We put 14 XSM motes in the area shown in Fig. 16 with the location indicated as in Fig. 17. We obtained 7 traces over an hour in the afternoon on three different days. To obtain the ground truth of event, we monitor the scene described in Section 3.6.

We determine *a hit* if the current detection energy is more than 6 times greater than an adaptive threshold, which is set to background noise. A XSM mote determines that an event occurs, if the number of hits during the last 10 consecutive sampling windows is greater than two. We use observation value 0 for event detection and 1 for no event detection. Similarity is calculated based on Equation (7) by setting $p = 1$ and T_{span} = $T_{update} = 100(sec)$. During T_{update}, observations are recorded, and at the end of round the similarity graph is updated. We used value $\alpha = 0.98$, $\beta = 0.05$ for similarity calculation. Based on the similarity between neighbors, a node determines its future sleeping schedule for each T_{sch} following the Algorithm 4. The compiled image of a mote implementation is 13,500 bytes of code memory and 532 bytes of data memory. Since the basic slot T_{sch} is 5(sec), motes need to be only loosely time synchronized

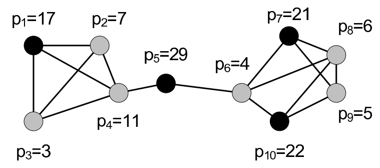


Fig. 15. Similarity Graph and Priority Comparison

Fig. 16. Experiments in the Study Area at Library

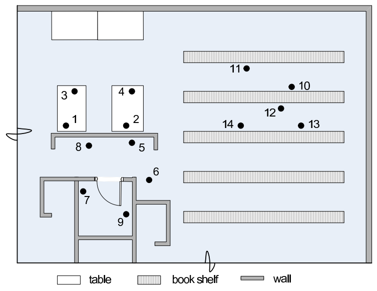


Fig. 17. Sensor Node Placement

among themselves. To achieve that, it is enough to use a lightweight NTP-Like time synchronization protocol, in which a mote broadcasts a message with its local time-stamp and its neighbor calculates the difference between the received timestamp and its local clock. More advanced time synchronization algorithms, such as FTSP [24], can also be used, if needed.

5 System Evaluation of SAM

We have described the implementation details of both P-SAM and V-SAM in Sections 3 and 4, respectively. In this section, we evaluate the effectiveness of our designs in various environments.

5.1 Evaluation on P-SAM Design

We provide implementation results in real-world settings as well as large scale simulation results.

5.1.1 Outdoor P-SAM Evaluation

In the P-SAM experiment, a person moved around a sensor sufficiently (10 times crossing straight over the area in different directions and positions). Figs. 18 and 19 show the sensing area we obtained after training a sensor, which is placed (1) in an open area and (2) in an area with a obstacle. The positions belonging to the detected events were associated to the closest grid points indicated in the figures. Fig. 18 indicates that the sensing area is irregular even without an obstacle. Fig. 19 shows that the obstacle affects the sensing area significantly. With the circle model (a disk with radius $400\ cm$), we expect a point within the circle to be associated with event detection and a point beyond the circle range not to be associated with event detection. After repeating the training test many times, we obtained irregularity and training confidence as shown in Table 1. They were calculated for all points associated with training events as follows:

$$\textbf{Irregularity} = \frac{n_1 + n_2}{n_3}$$

where n_1 is number of points inside the circle the events of which are not detected, n_2 is number of points outside the circle the events of which are detected, n_3 is number of points inside the circle.

$$\textbf{Confidence} = \frac{1}{number\ \ of\ \ points} \sum_{each\ \ point} MAX(p_1, p_2)$$

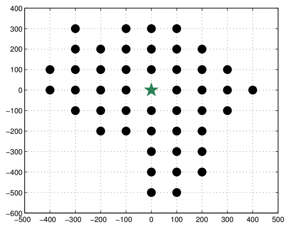


Fig. 18. Coverage without Any Obstacle in $1,000\ cm \times 1,000\ cm$ square

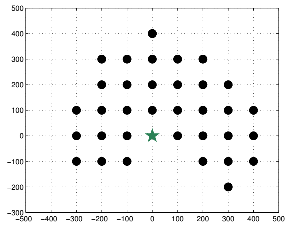


Fig. 19. Coverage with Obstacles in $1,000\ cm \times 800\ cm$ rectangle

TABLE 1
Sensing Area in Outdoor Experiment

Without obstacle		With obstacle	
Irregularity	Confidence	Irregularity	Confidence
0.367	0.83	0.387	0.80

where p_1 is fraction of detected events, p_2 is fraction of undetected events. Higher value of confidence means the same result is more likely to be reproduced as before.

We also performed experiments with MicaZ photo sensors in [17] where the data for training cost is provided. Note that 100 events of binary value can be transmitted with one small packet.

5.1.2 Simulation Setup

In the real environment, it is extremely difficult to obtain the ground truth of real sensing coverage. To overcome the limitation of outdoor experiments, we perform simulaions of P-SAM that incorporate knowledge of the ground truth in Sections 5.1.3 through 5.1.5.

We use an *oracle algorithm* that assumes knowledge of the sensing area of the nodes. We want to emphasize that the oracle algorithm and generated ground truth are used *only for the purpose of evaluation*. This knowledge is not used in any part of the P-SAM algorithm. The oracle generates a sensing pattern according to the following irregularity model, which is an extension of the DOI model [14].

$$R_\theta = \begin{cases} R_{min} + (R_{max} - R_{min}) \cdot Rnd & \theta = 0^\circ \\ R_{\theta-1} \pm Rnd \cdot var & 0^\circ < \theta < 2\pi \end{cases} \tag{13}$$

where R_{min} and R_{max} are the minimum and maximum coverage ranges respectively, and $R_\theta \in [R_{min}, R_{max}]$ is the sensing range at angle θ. Rnd is a random number

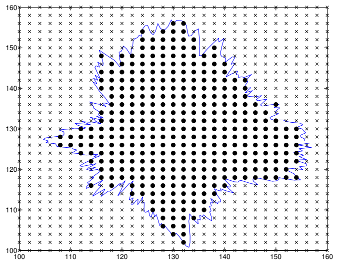


Fig. 20. Regular $G(t)$

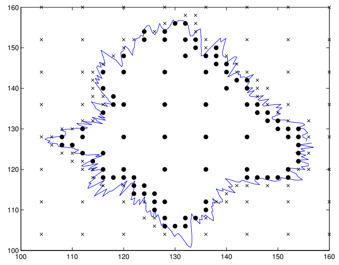


Fig. 21. Hierarchical $G(t)$

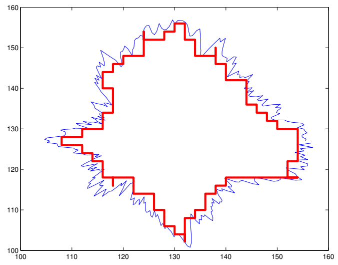


Fig. 22. Wrapping

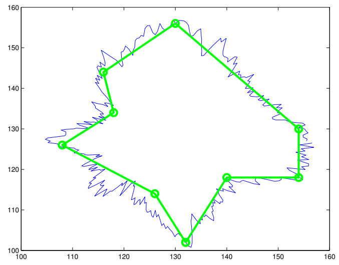


Fig. 23. Simplification

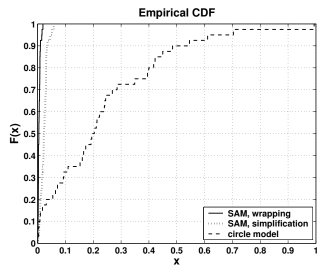


Fig. 24. The CDF fp curves for circular and two SAM models

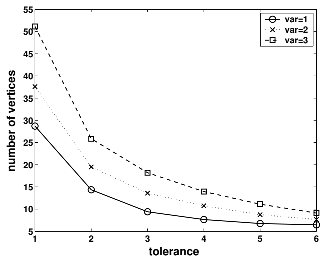


Fig. 25. Number of vertices with varying tolerance and irregularity

between 0 and 1, and var is a variation of the ranges at consecutive angles due to the irregularity.

We represented the deployment area with 512 by 512 square with 1400 nodes randomly placed. Starting from $R_{\theta_{min}}$ at $0°$, the real irregular coverage was generated for each sensor according to Equation (13) with $R_{min} = 10, R_{max} = 30$ and var = 2.0. The interval D was chosen from 2^i, where $1 \leq i \leq \lfloor log_2 R_{min} \rfloor$, so that $2^i < R_{min}$. In the regular training, the interval is fixed. However, in the hierarchical training starting from a certain initial interval $D = 2^i$ at level 1, the interval decreases to $2^{(i-1)}$ at level 2, and so on, until the smallest possible interval 2^j is reached at the last level $i - j + 1$.

5.1.3 A Case Study of P-SAM

In this case study, both the regular training and hierarchical training are evaluated as shown in Figs. 20 and 21, respectively. The continuous curve shown in these figures is the actual sensing area of a single node, which is unknown to SAM. Without polygon abstraction, a coverage for each sensor is represented by a set of event locations associated with value $S = 1$ (the solid circles shown in the figures are detected events). To simplify the representation, the sensing coverage is wrapped by boundary points on the grid. Fig. 22 shows the wrapping by grid points. The wrapping can be optimized in various ways. We use the DP algorithm for this purpose. When the training is fine grained the representation of sensing coverage in Fig. 22 requires upto hundreds of points even if we use only the boundary grid points. However, if we use the polygon simplification the representation reduces to several points as in Fig. 23.

5.1.4 Savings through Sensor Area Abstraction

Fig. 24 is the CDF curves of fp under three settings: circular model, P-SAM model with wrapping, and P-SAM model with simplification. We can clearly see that both P-SAM cases significantly outperform the circular model. The simplification approach uses less vertices to describe the area, thus it is slightly less accurate than the wrapping.

Fig. 25 shows that as the tolerance increases the number of vertices needed decreases. The variance of sensing coverage also affects the data abstraction. With the sensing irregularity of $var = 2.0$ and interval 1, the original sensing coverage is initially represented by raw data of 500–1500 vertices, including inner points of coverage. Using wrapping we can reduce this to 150–300 points, which might be still large for some memory constrained sensing devices. We optimize it further by polygon simplification. With tolerance $tol = 4$ (which is reasonable considering the value $R_{min} = 10, R_{max} = 30$), the sensing coverage can be simplificed as a polygon with only 5–15 vertices, which is about 99% reduction in data with slight decrease in accuracy.

5.1.5 Application Improvements Using P-SAM

We apply coverage scheduling based on individual sensor coverage by circular 0/1 model and by P-SAM. We vary node density from 0.958 to 6.707 following the commonly assumed node density in the existing works [4], [13], [21], [34]. The design goal of *full coverage scheduling* is to cover every physical point within an area with minimal energy consumption, and *point coverage scheduling* is to cover every target. The radius of a disk in circular model is denoted by R_c. Two key metrics for coverage applications are (i) **Fraction of Blind Areas** and (ii) **Energy Consumption**.

Fig. 26 shows the fraction of blind areas when different densities of nodes are scheduled by *full coverage scheduling*. As we increased the number of nodes from 200 to $1,400$, the blind area by P-SAM guided coverage scheduling significantly decreases. On the other hand, with an optimistic circular model (a disk with radius $R_c = 30$), the percentage of blind area stays at about 15%, despite the fact that over 1,400 nodes have been deployed into the area. Fig. 27 shows the average energy consumption per node. When a circular model is conservative, $R_c = 10$, the energy consumption remains the

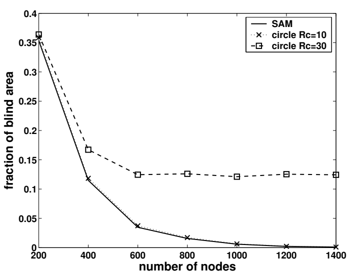


Fig. 26. Fraction of Blind Areas with Varying Densities

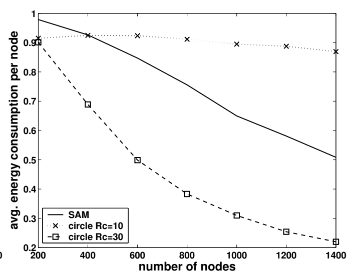


Fig. 27. Avg. Energy Consumed with Varying Densities

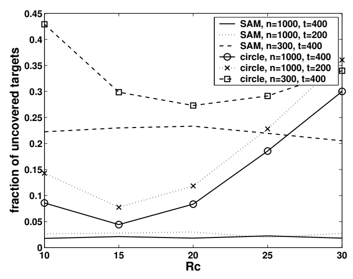


Fig. 28. Fraction of Uncovered Targets with Varying R_c

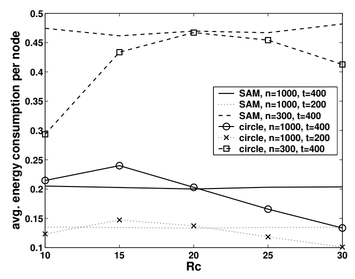


Fig. 29. Avg. Energy Consumed with Varying R_c

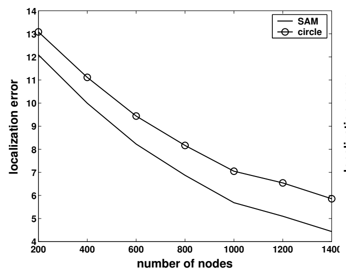


Fig. 30. localization error with varying Densities

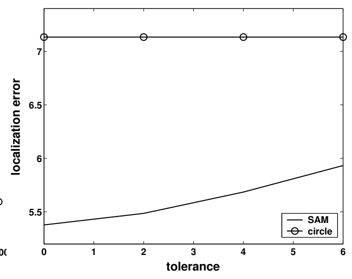


Fig. 31. localization error with varying tolerance

same for any density, while P-SAM has accurate sensing area information with a smaller energy consumption.

We also apply P-SAM and a circular model to the *art gallery* application where the sensor nodes are organized by *point coverage scheduling* to monitor a set of important stationary targets with known locations. Figs. 28 and 29 show the number of missing targets that are not covered by sensor nodes and the average energy consumption per node. The number of sensor nodes and the number of targets generated are denoted by n and t. As shown, the number of targets not covered in a circular model is larger than P-SAM. For example, when $n = 1,000, t = 400$, if we use P-SAM, the *point coverage algorithm* can cover every target. However, if we use the circular model with $R_c = 25$, it will miss about 20% of the targets.

The curves for the circular model shown in Figs. 28 exhibits an interesting $\bigcup$ shape. The number of missing targets can be reduced by decreasing R_c at the cost of increasing energy consumption. However, the coverage error cannot monotonically reduce forever. This is because if we reduce R_c into a small value, a node that can physically cover a target will mistakenly assume it cannot cover the target, and therefore turns itself off. This also explains why the energy consumption in circular model in Figs. 29 exhibits a $\bigcap$ shape.

Many tracking algorithms implemented on the sensor devices estimate the location of a sensed target by using the *centroid localization*. We apply P-SAM to the centroid localization. In centroid, if a set of sensors detects a target, the location of target (X_{est}, Y_{est}) is calculated by:

$$(X_{est}, Y_{est}) = \left(\sum_{i=1}^{N} X_i, \sum_{i=1}^{N} Y_i \right) \tag{14}$$

where $(X_i, Y_i), i = 1, \ldots, N$ is location of anchor beacon.

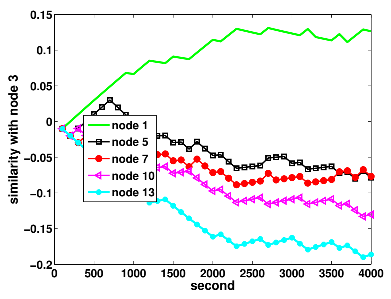
The assumption behind the centroid localization is that sensors have equal circular sensing ranges. Therefore, the location of the target is best estimated at the centroid of multiple sensors. With P-SAM, we know the sensing area of each sensor node which detects the targets. We can limit the location of target to the overlapping portion of these sensing areas. Then we can estimate the location of target by calculating the center of gravity of overlapping area. Fig. 30 compares the localization error between using circular model and using sensing area obtained by P-SAM according to different node densities. In Fig. 31, we change the tolerance tol used in simplification process of training coverage model. The coverage simplification with larger tolerance increases the localization error, but it is still below the localization error in circle coverage model even with large tolerance value $tol = 6$.

5.2 Evaluation on V-SAM Design

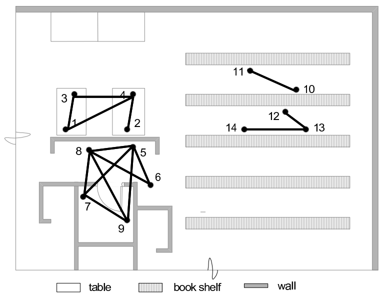
In this section, we evaluate the performance of V-SAM in the basement of university library. All experiments were conducted in an uncontrolled setting.

5.2.1 A Case Study of V-SAM

In this section, we provide snapshots of a V-SAM process. We ran 14 XSM motes for 4,000 seconds. During this period, two students were studying at a table, making small movements in the limited place, and there were several students passing the area. Fig. 32a shows the computed similarity between node 3 and nodes 1, 5, 7, 10, 13. As expected, the similarity between node 3 and node 1 always has positive values, while the similarity with other nodes has negative values. It is very interesting to observe that 1) at the beginning of the experiment, the pair-wise similarity values are about zero and that 2) with more observations over time, for the nodes with similar view similarity values increase, converging to a long–term positive value. Similarly for the nodes with different view, i.e., obstacles exist between them or they are far from each other, their similarity values decrease, converging to a long term negative value. Both long-term positive and negative values reflect the impact of environment, hardware and consistent movement for long time. And 3) the similarity fluctuates due to short-term behavior, because during the experiment, students



(a) Similarity between Node 3 and others



(b) Similarity Graph after 2000 seconds

Fig. 32. The Profile of Pair-wise Similarity over 4,000 seconds

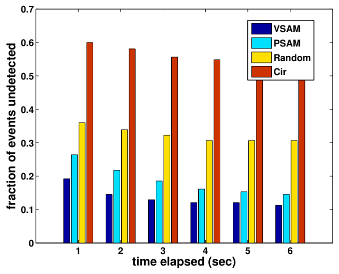


Fig. 33. Sensing Quality Comparison for Different Coverage Models

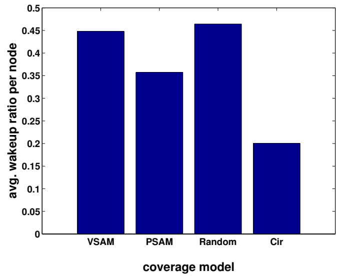


Fig. 34. Energy Consumption Comparisons for Different Coverage Models

made small random gestures. Fig. 32b is the similarity graph after 2000 seconds. We note that a mote can reside in multiple cliques in the similarity graph.

The cost to generate the similarity graph can be estimated by the number of messages for each node to generate. If each node generates a message whenever it detects an event within $T_{update} = 100\ (sec)$, the upper-bound on the number of messages generated by each node over $4,000$ seconds is 40. The average number of messages required by each node in this example is 13.1. However, if messages are all sent after $4,000$ seconds, each node only needs at most one message containing at most 5 bytes of data.

5.2.2 Application Improvements Using V-SAM

We compare the performance in coverage scheduling for four different coverage models: (i) V-SAM-guided coverage scheduling, (ii) random coverage scheduling, (iii) P-SAM-guided coverage scheduling, and (iv) circular model-guided coverage scheduling. We use two metrics to evaluate the coverage quality in our experiment: (i) **Fraction of Detection:** the percentage of detectable events that are actually detected, which is a metric to indicate the sensing quality, (ii) **Average Wakeup Ratio:** the average percentage of time that a node is awake, which is a metric to indicate the energy efficiency. To obtain the ground truth of events, we monitor the scene with a digital camcorder. We are especially interested in the detection of walking persons, a movement more than $2\ m$ is regarded as an event. To compare the sensing coverage under the same energy consumed, the wake-up schedule in random coverage scheduling is generated so that its average wake-up ratio is similar to the one in V-SAM guided coverage scheduling. For P-SAM-guided coverage scheduling, we represent sensing coverage with $1\ m$ interval. For circular-model-guided coverage scheduling, the sensing coverage is assumed as a circle with a radius of $6\ m$ centered around a mote's location. As a result, the energy consumption for circular model-guided coverage is much lower than for the others, as shown in Fig. 34.

The performance in sensing quality for four models (i) – (iv) is shown in Fig. 33. V-SAM has the lowest number of undetected movements, followed by P-SAM. Although P-SAM provides an accurate modeling method, it does not consider a behavioral model of events. For example, during the experiment, students sit next to one sensor and blocked its view temporally, which is not reflected in the training process in P-SAM. Even if the energy consumption in random coverage scheduling is adjusted slightly greater than V-SAM, its sensing quality is worse than V-SAM and P-SAM. For example, in Fig. 33, V-SAM-guided coverage scheduling has a greater number of events immediately detected in a second than the random case. Even after $6\ (sec)$ elapsed, the fraction of undetected events does not decrease in random coverage scheduling. In addition, as shown in Fig. 33, coverage scheduling based on a circular model misses more than half of detectable events. In V-SAM, we can better understand the similarity graph with a small T_{span}, resulting performance increase. However, without data pre-processing, i.e., dimensionality reduction, it may adversely cause performance decrease [17].

6 Discussion and Conclusion

This paper addresses sensing irregularity issues known but largely ignored by many designers. We contribute to this area by designing and implementing two complementary in-situ sensing modeling methods called P-SAM and V-SAM, respectively. By introducing controlled and monitored events, P-SAM provides accurate *sensing area models* for individual nodes. By utilizing natural events in the environments, V-SAM provides evolvable *sensing similarity models* automatically at low cost. Both models are generic enough to be used in many applications including sensing coverage and tracking. We have identified the impacts of sensing irregularity on typical applications as well as the improvements by using SAM.

Our design has been evaluated in test-beds consisting of 14 XSM motes in diversified environment settings including library and outdoor court yards.

Although we provided experiment results with PIR sensors in this paper, our solution can be applied to other type of sensors. Experiments with photo sensors are performend in [17]. Other sensors such as image sensor, acoustic sensor, magenetometer sensor can be trained with SAM without modification regardless of directionality of sensors, with events such as image, sound, deposits of iron – tank or airplane. Our approach will be especially useful to sensors having large coverage in the environment with lots of obstacles.

References

[1] Z. Abrams, A. Goel, and S. A. Plotkin. Set k-cover algorithms for energy efficient monitoring in wireless sensor networks. In *IPSN*, 2004.

[2] S. M. Alam and Z. Haas. Coverage and connectivity in three-dimensional networks. In *MobiCom*, 2006.

[3] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A wireless sensor network for target detection, classification, and tracking. *Computer Networks, Elsevier*, 2004.

[4] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic. Towards optimal sleep scheduling in sensor networks for rare-event detection. In *IPSN*, 2005.

[5] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. In *SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, 2001.

[6] V. Cevher and J. McClellan. Sensor array calibration via tracking with the extended kalman filter. In *Annual Federated Laboratory Symposium on Advanced Sensors*, 2001.

[7] CrossBow. *Product feature reference: sensor and functions*, 2003. http://www.xbow.com/Support/Support_pdf_files/Product_Feature_Reference%_Chart.pdf.

[8] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer 10 (2)*, 1973.

[9] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler. Design of a wireless sensor network platform for detecting rare, random, and ephemeral events. In *IPSN*, 2005.

[10] L. Girod, M. Lukac, V. Trifa, and D. Estrin. The design and implementation of a self-calibrating distributed acoustic sensing platform. In *SenSys*, 2006.

[11] B. Grabowski. Small robot sensors. http://www.andrew.cmu.edu/user/rjg/websensors/robot_sensors3.html.

[12] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, A. Tirumala, Q. Cao, T. He, J. A. Stankovic, T. Abdelzaher, and B. H. Krogh. Lightweight detection and classification for wireless sensor networks in realistic environments. In *SenSys*, 2005.

[13] C. Gui and P. Mohapatra. Power conservation and quality of surveillance in target tracking sensor networks. In *MobiCom*, 2004.

[14] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. Range-free localization schemes in large-scale sensor networks. In *MobiCom*, 2003.

[15] J. Hershberger and J. Snoeyink. Speeding up the douglas-peucker line-simplification algorithm. In *Proc 5th Symp on Data Handling*, 1992.

[16] C. Hsin and M. Liu. Network coverage using low duty-cycled sensors: random & coordinated sleep algorithms. In *IPSN*, 2004.

[17] J. Hwang, T. He, and Y. Kim. Exploring in-situ sensing irregularity in wireless sensor networks. In *SenSys*, 2007.

[18] F. Koushanfar, N. Taft, and M. Potkonjak. Sleeping coordination for comprehensive sensing using isotonic regression and domatic partitions. In *INFOCOM*, 2006.

[19] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: maximizing information while minimizing communication cost. In *IPSN*, 2006.

[20] KUBE Electronic LTD. *Optic TR230 PIR sensor*. http://www.kube.ch/downloads/pdf/kube_cone_optics_tr230.pdf.

[21] S. Kumar, T. Lai, and J. Balogh. On k-coverage in a mostly sleeping sensor network. In *MobiCom*, 2004.

[22] X. Y. Li, P. J. Wang, and O. Frieder. Coverage in wireless ad-hoc sensor networks. In *ICC*, 2002.

[23] J. Liu, J. Reich, and F. Zhao. Collaborative in-network processing for target tracking. *Journal on Applied Signal Processing*, 2002.

[24] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *SenSys*, 2004.

[25] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *INFOCOM*, 2001.

[26] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak. Exposure in wireless ad-hoc sensor networks. In *MobiCom*, 2001.

[27] V. Singhvi, A. Krause, C. Guestrin, J. Garrett, and H. S. Matthews. Intelligent light control using sensor networks. In *SenSys*, 2005.

[28] S. Slijepcevic and M. Potkonjak. Power efficient organization of wireless sensor networks. In *ICC*, 2001.

[29] R. Szewczyk, A. Mainwaring, J. Anderson, and D. Culler. An analysis of a large scale habit monitoring application. In *SenSys*, 2004.

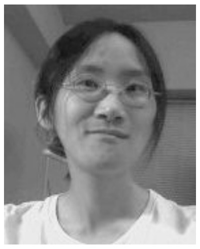
[30] D. Tian and N.D.Georganas. A node scheduling scheme for energy conservation in large wireless sensor networks. *Wireless Communications and Mobile Computing Journal*, 2003.

[31] G. Tolle, J. Polastre, R. Szewczyk, N. Turner, K. Tu, S. Burgess, D. Gay, P. Buonadonna, W. Hong, T. Dawson, and D. Culler. A macroscope in the redwoods. In *SenSys*, 2005.

[32] K. Whitehouse and D. E. Culler. Calibration as parameter in sensor networks. In *Workshop on Sensor Networks and Application (WSNA)*, 2002.

[33] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *SenSys*, 2004.

[34] T. Yan, T. He, and J. A. Stankovic. Differentiated surveillance for sensor networks. In *SenSys*, 2003.



Joengmin Hwang received her M.S. and Ph.D. degrees in Computer Science from the University of Minnesota - Twin Cities in 2004 and 2008, respectively. She received the B.E. degree from the Korea University, Seoul, Korea. Since 2008, she has been working as a research associate at the University of Minnesota - Twin Cities. She is author and co-author of publications in premier sensor network and security conferences and journals. Her research includes wireless networks, network security, embedded systems, mobile computing, and localization techniques.

Tian He is currently an assistant professor in Department of Computer Science and Engineering at the University of Minnesota - Twin Cities. He received the Ph.D. degree from the University of Virginia, Virginia in 2004, and the M.S. degree from the Institute of Computing Technology, Chinese Academy of Sciences, China, in 2000. He received his B.S. degree from the Nanjing University of Science & Technology, Nanjing, China in 1996. He is the recipient of the NSF career award and McKnight Land-Grant professorship from University of Minnesota in 2009.

Yongdae Kim received his B.S. and M.S. degrees in Mathematics from the Yonsei University, Seoul, Korea in 1991 and 1993, respectively, and Ph.D. degree in Computer Science from the University of Southern California in 2002. Since 2002 he has been on the faculty at University of Minnesota - Twin Cities. He received NSF career award on storage security and McKnight Land-Grant professorship award from University of Minnesota in 2005.