Exploring In-Situ Sensing Irregularity in Wireless Sensor Networks *

Joengmin Hwang, Tian He, Yongdae Kim

Department of Computer Science and Engineering
University of Minnesota, Minneapolis

jhwang@cs.umn.edu, tianhe@cs.umn.edu, kyd@cs.umn.edu

Abstract

The circular sensing model has been widely used to estimate performance of sensing applications in existing analysis 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 existence of obstacles and sensing irregularity introduced by 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. P-SAM provides accurate *sensing area models* for individual nodes using controlled or monitored events, while V-SAM provides continuous *sensing similarity models* 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 three testbeds consisting of 40 MICAz motes and 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

Categories and Subject Descriptors

I.6.5 [**Simulation and Modeling**]: Model Development—*modeling methodologies*

General Terms

Measurement, Performance, Design, Experimentation

Keywords

Model, Irregularity, Event, Sensing, Coverage, Scheduling, Similarity

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

SenSys'07, November 6–9, 2007, Sydney, Australia.

1 Introduction

Wireless sensor networks are envisioned to support variety of applications such as military surveillance [3, 18, 26], habitat monitoring [4, 7, 32], infrastructure protection [37] and scientific exploration [34]. As a bridge to the physical world, sensing is an indispensable elements 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 [9, 14, 15, 22], but researchers still continue to develop, simulate, and analyze sensor network protocols that utilize a simplified theoretical sensing coverage model [1, 6, 10, 16, 19, 23, 24, 31, 33, 35, 38], 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 environments. 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-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 some 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 a network of 14 XSM motes, 40 MICAz 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 significant 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 the indoor and outdoor system evaluations of both P-SAM and V-SAM. 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, 25, 28, 29] 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, 6, 16, 19, 23, 31, 33, 38] 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 [5, 8, 13, 36] tried to calibrate real sensing patterns to the standardized units. For example, sensor array calibration based on constant target tracking was proposed in [8]. The concept of macro-calibration for localization was introduced in [36]. Auto-calibration for acoustic sensor network was designed and implemented in [13]. 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 degrade 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 [20, 21, 30] without any prior assumptions on the sensing coverage. Koushanfar et al. [20] 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. The goal is to create the maximal number of uncorrelated subgroups, so that energy can be saved by turning on only one subgroup at a time. Krause et al.[21] 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 achieving the required performance. 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, 6, 16, 19, 23, 31, 33, 38]

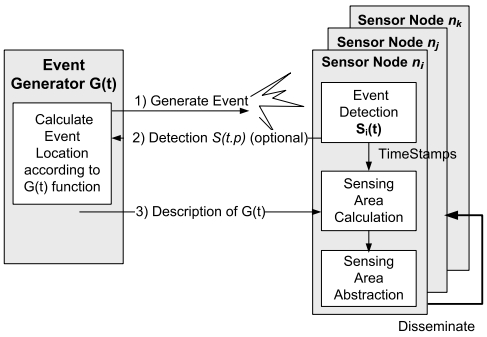


Figure 1. P-SAM Architecture

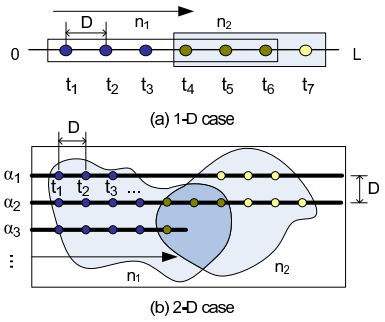


Figure 2. Regular Training

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 [32, 34]. We also assume the event type is known. This assumption is needed because the sensing area we obtain for one event type (e.g., movement of vehicles) cannot be applied to other types of events (e.g., light). If a network is designed to detect several types of events, sensing modeling for each type is required. Without loss of generality, we first describe our design as conceptually independent of the type of events used. Later on, we use light and Passive InfraRed (PIR) motion sensors as specific examples in indoor and outdoor P-SAM implementation, respectively.

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 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 [38] 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: $G : \mathbb{R} \rightarrow \mathbb{R}^2$, where $G(t) = p_t = (x_t, y_t)$ and $t \in \{t_1, t_2, ..., t_n\}$. In case of continuous events, a set of discrete training events can be obtained by sampling a continuous event with a certain interval.

Figure 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 Figure 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 stamps into $G(t)$, a set of timestamps $T_i = \{t_1^i, t_2^i, \ldots, t_n^i\}$ from node n_i can be converted to a set of locations $P_i = \{p_1^i, p_2^i, \ldots, p_n^i\}$. 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. There is a trade-off between the number of training events and the details of the coverage shape we obtain.

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 $T_i = \{t_1^i, t_2^i, \ldots, t_n^i\}$
10: P_i is a set of positions p where $S_i(t,p) = 1$

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 Figure 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

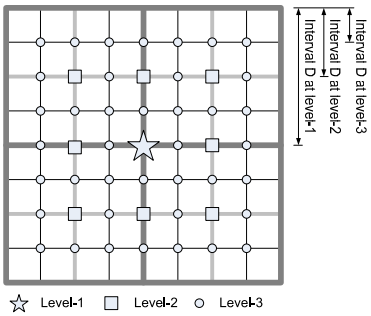


Figure 3. Hierarchical Partition

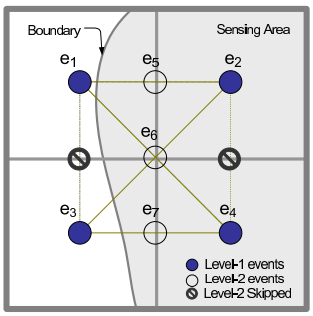


Figure 4. Level of Details

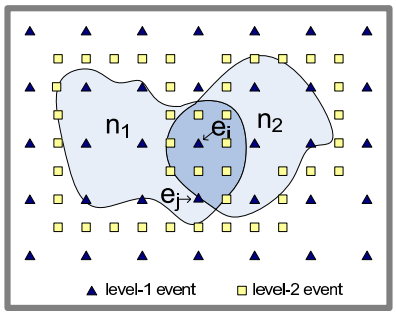


Figure 5. Hierarchical Training

$[0,L]$ with constant speed v with an *interval* D. Formally, $G(t) = t \cdot v$, where $t = kD/v$ and $0 \leq k \leq L/D$. For example, in Figure 2a, a sensor node n_1 collects a set of six timestamps $T_1 = \{t_1, t_2, \ldots, t_6\}$ at which the events are detected. Using function G, the actual locations of events are converted to a set of locations $P_1 = \{t_1 v, t_2 v, \ldots, t_6 v\}$. The sensing coverage of sensor n_1 can be defined as the line segment that covers P_1. Sensor n_2 reports timestamps $T_2 = \{t_4, t_5, t_6, t_7\}$ and the sensing coverage of sensor n_2 is defined as the line segment that covers $P_2 = \{t_4 v, t_5 v, t_6 v, t_7 v\}$. The intersection of T_1 and T_2, $T_1 \cap T_2 = \{t_4, t_5, t_6\}$ indicates that the coverage of the two sensors overlap, as shown in Figure 2a.

The regular training can be generalized to the case when the events occur in a plane. Figure 2b shows this approach. In this case, training area A is divided into 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 $S(t,p)$. In regular $G(t)$, it is desirable to cover every point in the area at least once. 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 Figure 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 Figure 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.

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 $S_i(t,p) = 1$

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. Figure 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 located at the middle of selected pairs of adjacent events at times t_5, t_6, and t_7, as shown in Figure 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

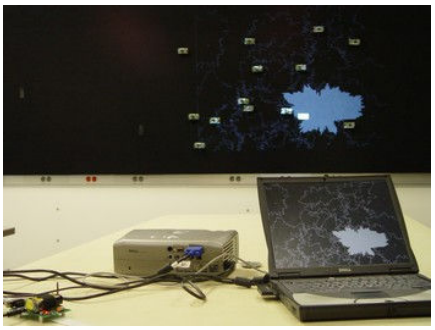


Figure 6. P-SAM System Setup

Figure 7. Hierarchical G(t) Scenario

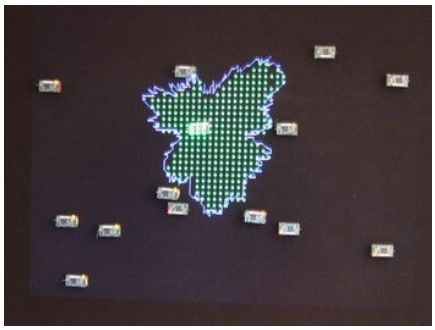


Figure 8. Sensing Area Obtained

Algorithm 3 P-SAM Coverage scheduling [38] 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}.

$e_5 \leftrightarrow e_1$, 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. Similarly, a coarse shape of sensing coverage is exposed and refined with a high level of detail in the boundary area. In a multiple nodes case, 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. Figure 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. Recursively, more level-2 events are generated on the boundary area of the sensing coverage, as shown in Figure 5.

3.3 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

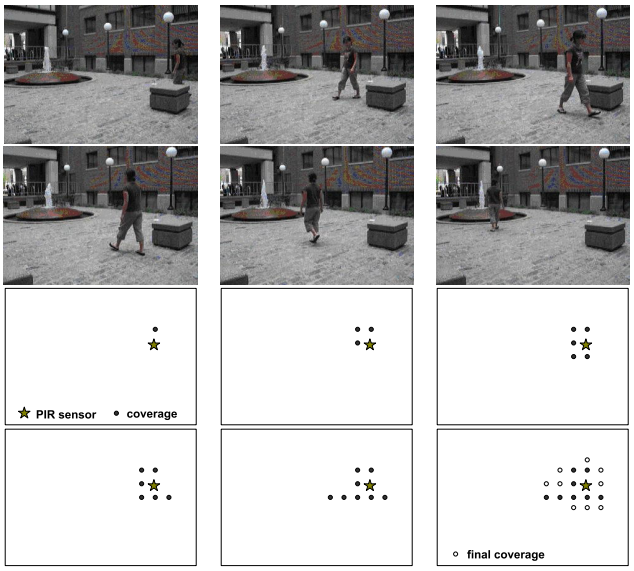


Figure 9. Mapping Event Detection to the Event Position Using Image Capture at Time $t = 6, 8, 10, 16, 17, 20$ (sec); last map Includes Additional Training Results by $t = 40$ (sec)

scheduling algorithm proposed by [38] to show its effectiveness.

Algorithm 3 describes how the coverage scheduling algorithm in [38] 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 to Algorithm 3.

3.4 Implementation of P-SAM

We have implemented the P-SAM system both in indoor and outdoor environments. These two implementations allow us to investigate several sensing modalities and different event control techniques at various kinds of environmental settings.

3.4.1 Indoor P-SAM system

We design and implement an indoor P-SAM system that includes regular and hierarchical training on the TinyOS/Mote platform. NesC [12] language is used to program the motes, and Java is used to build the regular and hierarchical generators. The compiled image of a full mote implementation occupies 14,500 bytes of code memory and 605 bytes of data memory. As shown in Figure 6, we attach 40 MICAz motes on a veltex black board and use a projector to generate regular and hierarchical events. The location of these events can be optionally displayed on the board. For example, we can visually inspect the distribution of hierarchical events as shown in Figure 7. For each generated event, we assign a unique ID. By using these IDs, we eliminate the need for time synchronization. After each run, the training results are visualized on the board and compared with the ground truth, as shown in Figure 8.

3.4.2 Outdoor P-SAM system

In the outdoor P-SAM system, we use ExScal XSM motes [11] 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 Figure 2b, we used monitored events (i.e., natural movements of a person). To map the event time to the event position, we used a digital 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 Figure 9 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.

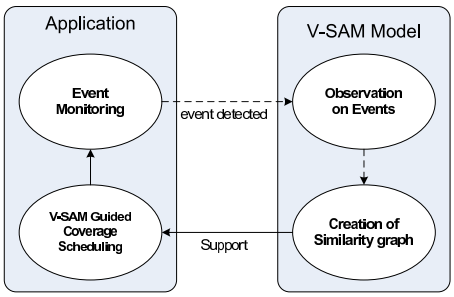


Figure 10. V-SAM and Coverage Scheduling Built-Upon It

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 especially in an area with unknown obstacles.

4.1 Main Idea

Figure 10 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.

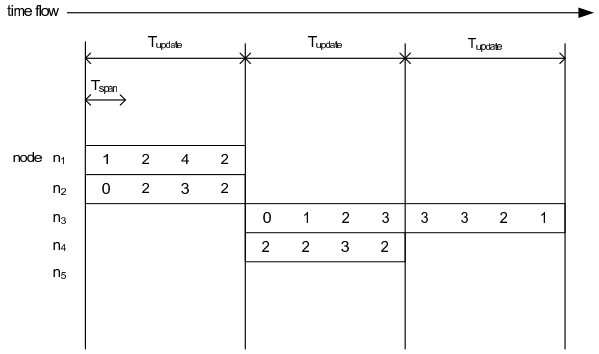


Figure 11. Time Series of Event Observation

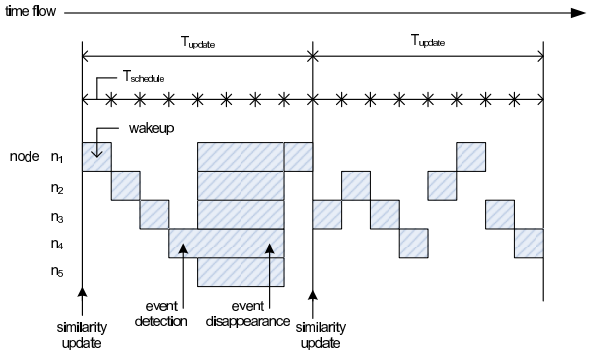


Figure 12. V-SAM Guided Coverage Scheduling

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 Figure 11, nodes are roughly synchronized with each other [27], 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:

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

where $\mathcal{P}$ can be $1, 2, \ldots, \infty$. This similarity is transformed by $d(i,j) \leftarrow (-2) \times \frac{d(i,j)-d_{min}}{d_{max}-d_{min}} + 1$ so that it is distributed between -1 and 1. *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:

$$\hat{d}^n(i,j) = \begin{cases} \text{if event is detected} \\ \quad \alpha \times \hat{d}^{n-1}(i,j) + (1-\alpha) \times d^n(i,j) \\ \text{otherwise} \\ \quad \beta \times \hat{d}^{n-1}(i,j) \end{cases} \tag{2}$$

We provide an example in Figure 11. 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 (3)-(4) are satisfied.

$$R^t(i,j) \leftarrow \text{Rnd}(i \,||\, j \,||\, t) \tag{3}$$

$$R^t(i,j) \;<\; w \cdot \hat{d}^n(i,j) \tag{4}$$

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:

$$\Pr[\, R^t(i,j) \;\leq\; w \cdot \hat{d}^n(i,j) \,] \tag{5}$$

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.

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 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 wake-up 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.

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 Figure 12, 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 (3)-(4) [Line 4-8]. For each node n_i, we define the priority of node i at time t as

$$p_i^t = \mathrm{Rnd}(i \,||\, t) \,||\, i. \tag{6}$$

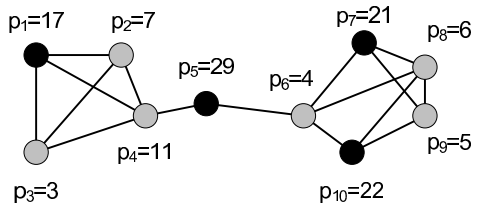


Figure 13. Similarity Graph and Priority Comparison

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, Figure 13 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 described in Section 4.2.2.

4.3.2 Sensing Quality and Coverage Scheduling

The basic algorithm introduces a way to avoid unnecessary energy consumption. We can further extend the basic algorithm, so that we can control the sensing quality as desired. For example, we can provide redundant or partial cov-

Figure 14. Experiments in the Study Area at Library

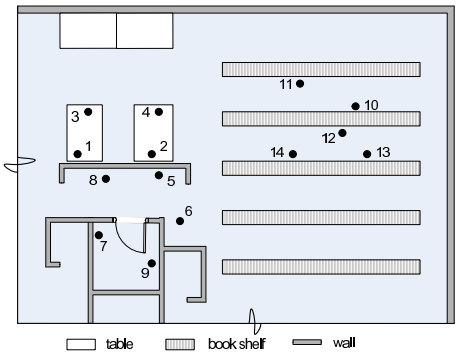


Figure 15. Sensor Node Placement

erage, coverage with a certain detection delay, and so on. To control the sensing quality in V-SAM, we propose two methods.

Neighbor-hood Control: By expanding the neighbor-hood, we can select fewer sensor nodes on the similarity graph. In the basic design, the priority values of one-hop neighboring nodes are compared. To reduce the sensing quality, each sensor node n_i compares its priority with node n_j within n-hop instead of with only one-hop neighboring nodes.

Rank-Based Control: To provide a high quality of sensing coverage, we can choose more than one sensor node during the priority comparison. A sensor node is chosen to wake up if its priority is ranked within top n among its 1-hop neighbor on graph G.

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 Figure 14. 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 Figure 14 with the location indicated as in Figure 15. 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 3.4.2.

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 (1) by setting $p = 1$ and $T_{span} = T_{update}$. 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 about 5 to 10 seconds, motes need to be only loosely time synchronized 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 timestamp and its neighbor calculates the difference between the received timestamp and its local clock. More advanced time synchronization algorithms, such as FTSP [27], 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 Outdoor P-SAM Design

In the outdoor P-SAM experiment, a person moved around a sensor sufficiently (10 times crossing straight over the area in different directions and positions). Figures 16 and 17 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. Figure 16 indicates that the sensing area is irregular even without an obstacle. Figure 17 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.

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

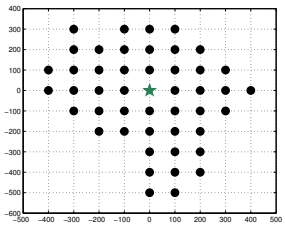


Figure 16. Coverage without Any Obstacle in $1,000\ cm \times 1,000\ cm$ **square**

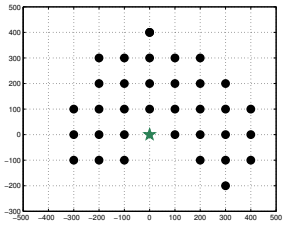


Figure 17. Coverage with Obstacles in $1,000\ cm \times 800\ cm$ **rectangle**

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.

5.2 Indoor P-SAM Evaluation

In an outdoor environment, it is extremely difficult to obtain the ground truth of real sensing coverage. Without ground truth, we can only investigate the characteristics of sensing coverage. To overcome the limitation of outdoor experiments, in this section we extend the evaluation of P-SAM by incorporating knowledge of the ground truth in a controlled indoor environment.

5.2.1 Ground Truth

We use an *oracle algorithm* that assumes knowledge of the sensing area of the nodes. Basically, this algorithm activates a sensor node (e.g., through projecting light to a sensor shown in Figure 8) if the controlled event $e(t,p)$ is within the sensing area of the node. 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 [17].

$$R_\theta = \begin{cases} R_{min} + (R_{max} - R_{min}) \cdot Rnd & \theta = 0^\circ \\ R_{\theta-1} \pm Rnd \cdot var & 0^\circ < \theta < 2\pi \end{cases} \quad (7)$$

where R_{min} is the minimum coverage range, R_{max} is the maximum possible coverage range, and $R_\theta \in [R_{min}, R_{max}]$ is the sensing range at angle θ. *Rnd* is a random number between 0 and 1, and *var* is a variation of the ranges at consecutive angles due to the irregularity. With a higher value of *var*, we introduce more irregularity.

5.2.2 System Implementation and Setup

We designed and implemented a complete version of training that includes regular and hierarchical training on the TinyOS/Mote platform. We attached 40 MICAz motes on a veltex black board and used a projector to generate regular and hierarchical events. We represented the deployment area with 128 by 128 square with 10 to 40 micaZ motes randomly placed. Starting from $R_{\theta_{min}}$ at 0°, the real irregular coverage was generated for each sensor according to Equation (7) with $R_{min} = 10, R_{max} = 30$ and *var* = 1.0, 2.0 or 3.0 (default is 2.0). The interval D was chosen from 2^i, where $1 \le i \le \lfloor log_2 R_{min} \rfloor$, 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 $i - j + 1$.

5.2.3 Evaluation Metrics

We define two types of coverage error: (i) **false positive rate** fp**:** The number of points in sensing area obtained by P-SAM but not in sensing area of the ground truth, divided by the number of points in sensing area of the ground truth. (ii) **false negative rate** fn**:** The number of points in sensing area of the ground truth but not in sensing area obtained by P-SAM, divided by the number of points in sensing area of the ground truth.

5.2.4 fp and fn of Sensing Coverage

Coverage error increases under the following two conditions (i) the irregularity of sensing area increases, or (ii) granularity of training data (interval D defined in Section 3.2.1 and 3.2.2) becomes larger. In regular training, the event layouts generated are grids with different intervals (from 1 to 4). In hierarchical training, we use the same initial interval, but different last-level intervals. Figure 18 shows that with a small interval, we can achieve very precise coverage modeling. fp is almost 0% and fn is at 1% to 8%. The coverage error in Figure 18 for a certain fixed interval in the regular training is very similar to the coverage error in Figure 19 for the corresponding last level interval in the hierarchical training. In the hierarchical training, changes in the initial interval make no difference in coverage error as long as the last level interval is the same.

5.3 Application Improvements Using P-SAM

In evaluation, we apply coverage scheduling based on individual sensor coverage by circular 0/1 model and by P-SAM. The evaluation is done by simulation with the same setting as in Section 5.2.2, except with a larger area (512 by 512 square) and with more sensor nodes. We vary node den-

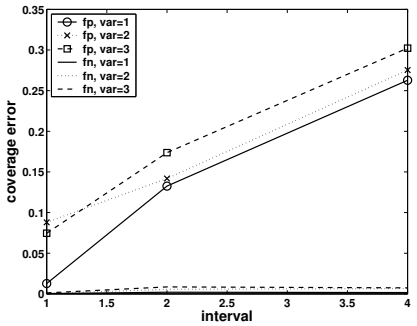


Figure 18. Errors in Regular $G(t)$ with Varying Intervals and Irregularity

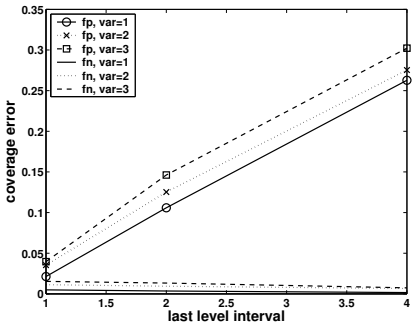


Figure 19. Errors in Hierarchical $G(t)$ with Varying Intervals and Irregularity

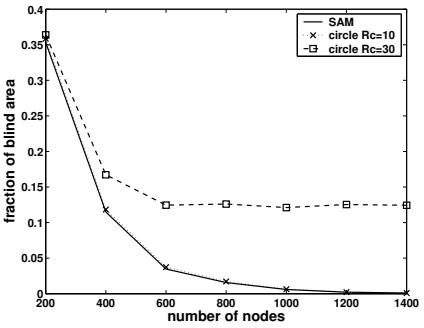


Figure 20. Fraction of Blind Areas with Varying Densities

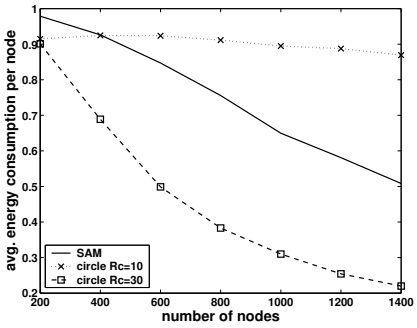


Figure 21. Avg. Energy Consumed with Varying Densities

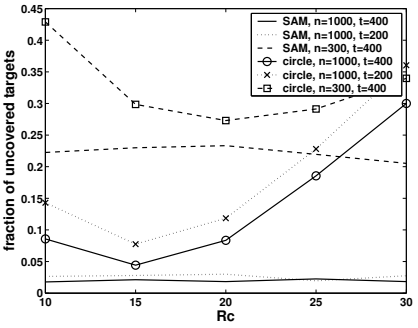


Figure 22. Fraction of Uncovered Targets with Varying R_c

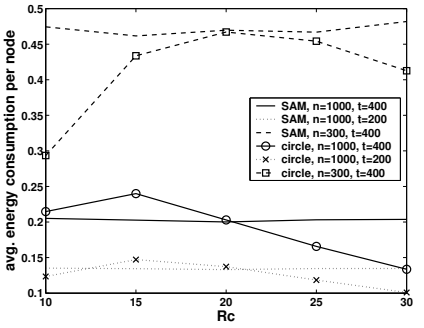


Figure 23. Avg. Energy Consumed Per Node with Varying R_c

sity from 0.958 to 6.707 following the commonly assumed node density in the existing works [6, 16, 23, 38].

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**.

Figure 20 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. Figure 21 shows the average energy consumption per node. When a circular model is conservative, $R_c = 10$, the energy consumption remains the same for every different 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. Figures 22 and 23 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 Figure 22 and 23 exhibit 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 exhibits a $\bigcap$ shape.

5.4 System Evaluation of V-SAM

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

5.4.1 Case Study

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. Figure 24 shows the record of event

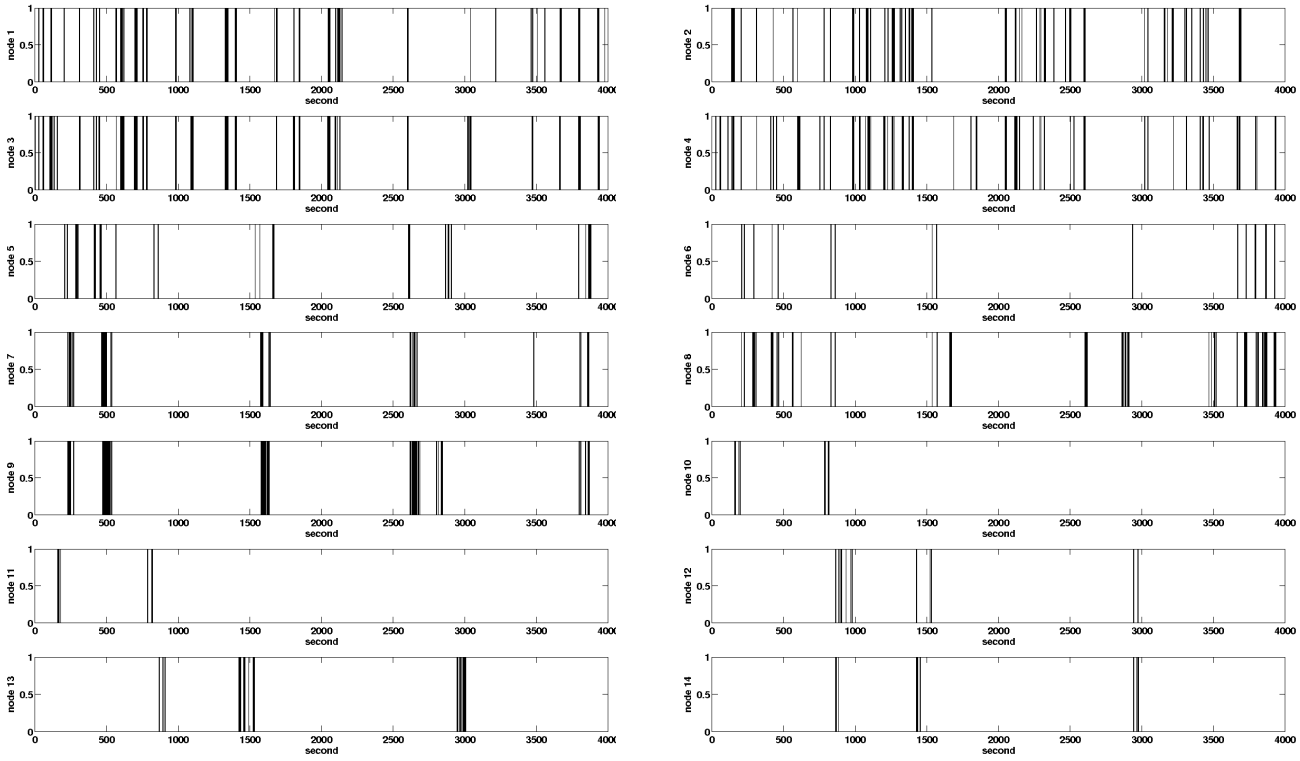


Figure 24. Event Detection Profile of 14 XSM Nodes over 4,000 seconds on the Ground Floor of the University Library

detection for each sensor. Frequent event detection by sensor nodes 1 - 4 was due to the small consistent movements by two students. The rest of the sensor nodes reflect a few movements of passing people. Obviously, even with the visual view of the record, it is easy to find that motes (1,2,3,4) have similar views. Similarly, we can group motes (5,6,8), motes (7,9), motes (10, 11), and motes (12,13,14). The similarity is continuously computed on each mote. Figure 25 shows the computed similarity between nodes and similarity graph as the time passed. To make Figure 25 readable, we present only the similarity values between a node and other five nodes (one from each group).

As expected, Figure 25 indicates that the intra-group similarity always has positive values, while the inter-group similarity 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, the intra-group similarity values increase, converging to a long–term positive value. Similarly inter-group 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 made small random gestures. We note that a mote can reside in multiple groups in the similarity graph. For example, in Figures 25b and 25c, we can observe that mote 5 sometimes has positive similarity values with mote 7 in other group. This multi-group similarity makes simple group-based rotation ineffective, which explains why the coverage algorithm proposed in Section 4.3 is needed.

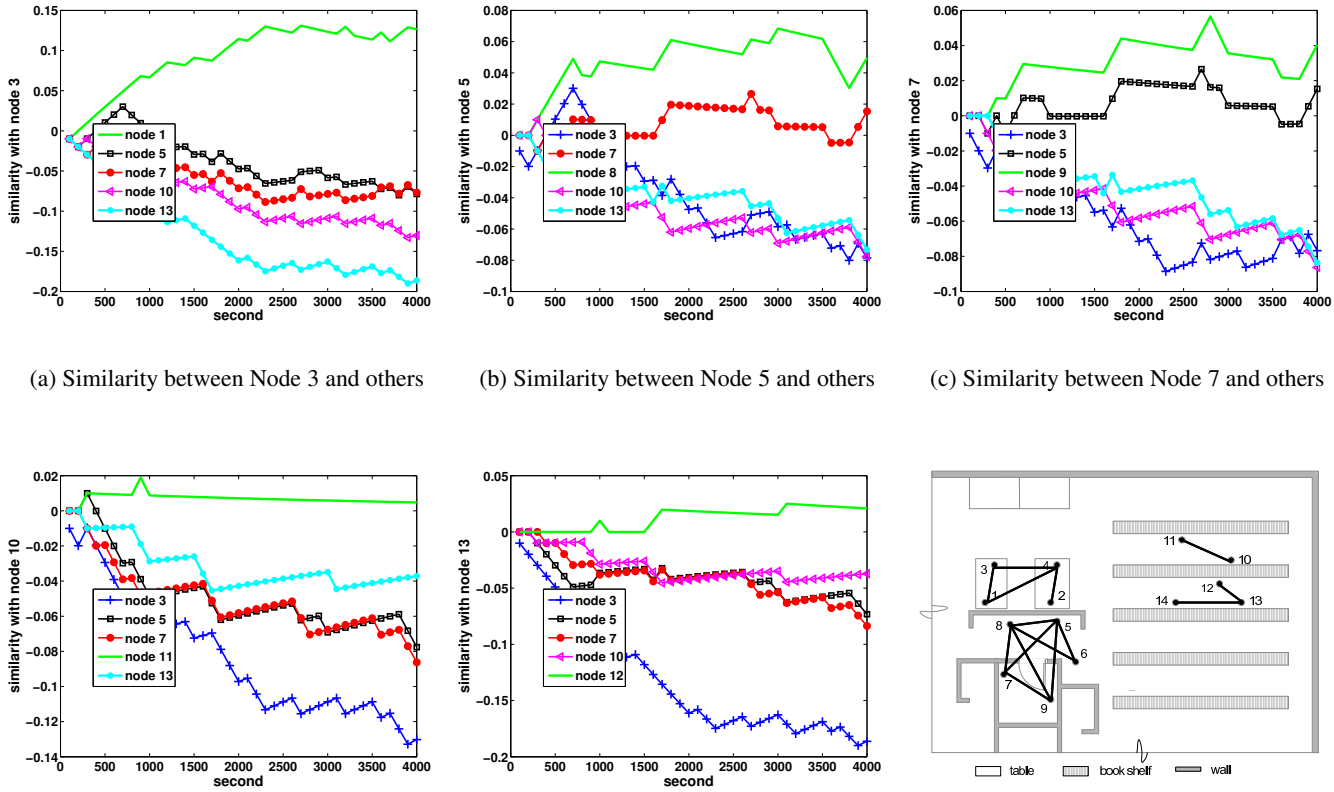
The cost to generate the similarity graph can be estimated by the number of messages for each node to generate. If each node generates a message whenever it detects an event within $T_{update} = 100\ (sec)$, the upperbound on the number of messages generated by each node over $4,000$ seconds is 40. The 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.

5.4.2 Evaluation Metrics

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.

5.4.3 Coverage under different T_{update}, T_{sch} Settings

We compare the coverage achieved in V-SAM guided coverage scheduling and random coverage scheduling algorithm. 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, we generate the wake-



(a) Similarity between Node 3 and others

(b) Similarity between Node 5 and others

(c) Similarity between Node 7 and others

(d) Similarity between Node 10 and others

(e) Similarity between Node 13 and others

(f) The Similarity Graph after 2000 seconds

Figure 25. The Profile of Pair-wise Similarity over 4,000 seconds

up schedule in random coverage scheduling so that its average wake-up ratio is the same as in V-SAM guided coverage scheduling.

Varying a value of T_{update}, T_{sch}, we obtained the result shown in Figures 26 to 28. Overall, for the fixed energy consumption, the V-SAM-guided coverage scheduling has better performance than the random coverage scheduling. For example, in Figure 26, 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, while it decreases in V-SAM-guided coverage scheduling. Figure 27 shows the similar performance when $T_{sch} = 10$ (sec). With slow change in wakeup schedule with $T_{sch} = 10$ (sec), the performance degrades more than a faster change with $T_{sch} = 5$ (sec). This is because a fast switch among sensor nodes leads to a shorter delay in detection, which confirms the results reported in [6, 16].

In V-SAM-guided coverage scheduling, the value T_{update} should be selected properly to make the coordination effective for detecting persons' walking. Figure 28 shows the fraction of undetected events when we set T_{update} to 5 (sec). The number of undetected events are smaller than the case with $T_{update} = 100$ (sec) in Figure 26. However, in Figure 29, a sensor's average wake-up ratio is 75% when $T_{update} = 5$ (sec), which is almost 1.5 times more than an 47% wake-up ratio for the case $T_{update} = 100$ (sec). This is because with $T_{update} = 5$ (sec), the similarity measure for a person's walking is prone to a noise such as small gestures, leading to sensor nodes uncorrelated and causing higher wake-up ratio.

5.4.4 *Performance Comparison*

In this section, 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. 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 P-SAM-guided coverage scheduling, is similar to V-SAM-guided or random coverage scheduling, while the energy consumption for circular model-guided coverage is much lower than for the others, as shown in Figure 31.

The performance in sensing quality is shown in Figure 30.

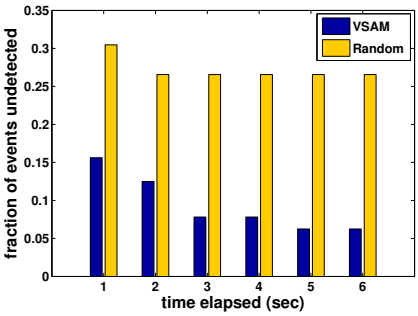


Figure 26. $T_{update} = 100\ (sec)\ \ T_{sch} = 5\ (sec)$

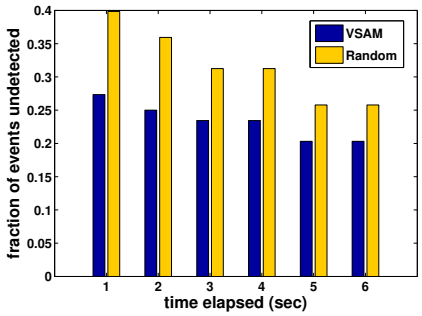


Figure 27. $T_{update} = 100\ (sec)\ \ T_{sch} = 10\ (sec)$

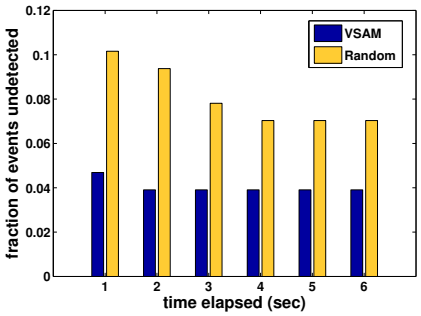


Figure 28. $T_{update} = 5\ (sec)\ \ T_{sch} = 5\ (sec)$

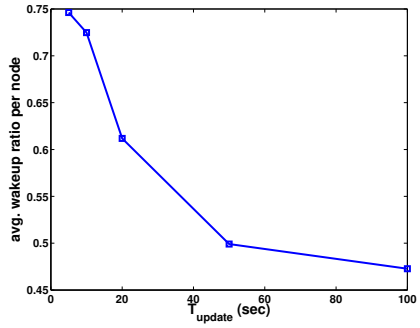


Figure 29. Energy Consumption for Different Settings of T_{update}

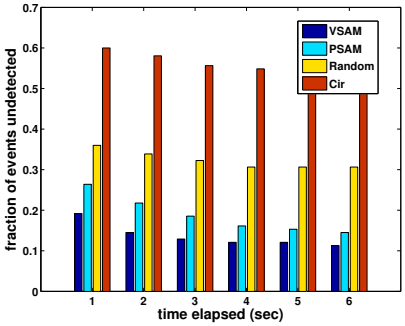


Figure 30. Sensing Quality Comparison for Different Coverage Models

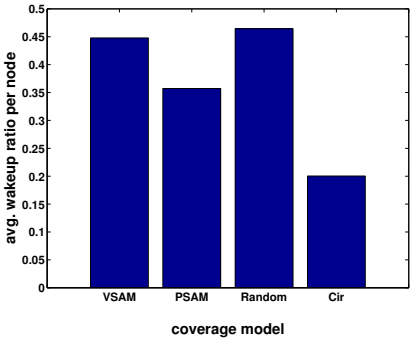


Figure 31. Energy Consumption Comparisons for Different Coverage Models

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. In addition, as shown in Figure 30, coverage scheduling based on a circular model misses more than half of detectable events.

6 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. Our design has been evaluated in three testbeds consisting of 40 MICAz motes and 14 XSM motes. We have evaluated our system extensively in diversified environment settings including indoor controlled labs, uncontrolled library and outdoor court yards. In addition, we have identified the impacts of sensing irregularity on typical applications as well as the improvements by using SAM. We hope this work motivates our community to seriously consider the in-situ sensing phenomena in the sensor networks.

7 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] M. A. Batalin, M. Rahimi, Y. Yu, D. Liu, A. Kansal, G. S. Sukhatme, W. J. Kaiser, M. Hansen, G. J. Pottie, M. Srivastava, and D. Estrin. Call and response:Experiments in sampling the environment. In *SenSys*, 2004.

[5] V. Bychkovskiy. Distributed in-place calibration in sensor networks. In *M.S. Thesis, University of California, Los Angeles*, 2003.

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

[7] 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.

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

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

[10] O. Dousse, P. Mannersalo, and P. Thiran. Latency of wireless sensor networks with uncoordinated power saving mechanisms. In *MobiHoc*, 2004.

[11] 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.

[12] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC language: A holistic approach to networked embedded systems. In *PLDI*, 2003.

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

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

[15] 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.

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

[17] 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.

[18] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh. Energy-efficient surveillance system using wireless sensor networks. In *MobiSys*, 2004.

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

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

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

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

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

[24] S. Kumar, T. H. Lai, and A. Arora. Barrier coverage with wireless sensors. In *MobiCom*, 2005.

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

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

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

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

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

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

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

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

[33] 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.

[34] 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.

[35] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *SenSys*, 2003.

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

[37] 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.

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