Realistic Sensing Area Modeling

Joengmin Hwang, Yu Gu, Tian He, Yongdae Kim
Department of Computer Science, University of Minnesota, Minneapolis
{jhwang,yugu,tianhe,kyd}@cs.umn.edu

***Abstract*—Despite the well-known fact that sensing patterns in reality are highly irregular, researchers continue to develop protocols with simplifying assumptions about the sensing. For example, a circular 0/1 sensing model is widely used in most existing simulators and analysis. While this model provides high-level guidelines, it could cause wrong estimation of system performance in the real world. In this project, we design and implement a practical Sensing Area Modeling technique, called SAM. By injecting events through regular and hierarchical training, SAM estimates the sensing areas of individual sensor nodes accurately. Especially, this work is the first to investigate the impact of irregular sensing area on application performance, such as coverage scheduling. We evaluate SAM using outdoor experiments with XSM motes, indoor experiment with 40 MicaZ motes as well as an extensive 1000-node simulation. Our evaluation results reveal serious problems caused by circular sensing model, while demonstrating significant performance improvements in major applications when SAM is used.**

I. Introduction

As a bridge to the physical world, sensing is indispensable for many sensor network systems, such as military surveillance [1], habitat monitoring [2], infrastructure protection [3] and scientific exploration [4]. Compared with diversified solutions for communication among sensor nodes, research on sensing is still premature. One well-known, but largely ignored, issue is the sensing irregularity. It has been known for years that the sensing pattern is not regular, but researchers still continue to develop, simulate and analyze sensor network protocols, assuming a circular 0/1 sensing model, i.e., the sensing boundary is represented by a circle (a sphere in 3D) centered at a sensor. We acknowledge that the results based on this simplifying assumption or its derivations can reveal good insights, but they often lead to an all-too-common problem found today where solutions developed by simulation and analysis do not work in the real world. Our work is motivated by the fact that it is difficult to describe the realistic sensing coverage through theoretical modeling. For example, at the time of manufacturing, calibration might not be accurate enough, introducing heterogeneity among the same type of sensing devices. Even if it is possible to precisely calibrate the sensors, environmental impact (e.g., obstacles) can severely affect the sensing characteristics, causing irregular and non-uniform sensing patterns at different sensor nodes. Since irregularity is a common issue in sensor networks, therefore, it is unwise for the developers continue to ignore such reality, blindly assuming the circular sensing model.

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.

Our answer to this issue is a sensing area modeling technique called SAM, which obtains the coverage of sensor nodes through event training. The main idea is to identify the sensing coverage based on event detection results by individual sensor nodes. A key architectural advantage of this approach is a lightweight design in sensor node with minimal overhead. Besides communication, each sensor node only needs to support a simple detection function. Specifically, our contributions in this work lie in:

- **Modeling and Validation:** We propose and implement regular and hierarchical training-based modeling approaches. We validate the accuracy of our modeling approach using outdoor experiment with XSM motes, indoor experiment with 40 MicaZ motes as well as an extensive 1000-node simulation.
- **Impact Analysis and Solutions:** Our model serves two research purposes. First, SAM enhances 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 first to study the impact of sensing irregularity on application performance such as coverage scheduling algorithms. 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 as follows. Section II outlines the SAM design, followed by a detailed modeling design in Section III. We present outdoor experiment in Section IV and indoor emulation in Section V. Section VI shows application performance improvements when SAM is used. Section VII concludes the paper.

II. The Overview of the SAM Design

In this section, we introduce the design of SAM at the architectural level. We target to static sensor networks (no mobility), which is the case for most existing deployed sensor systems [2], [4]. We assume the type of events is known. This assumption is needed because the sensing area we obtain for one type of events (e.g., vehicles) cannot be applied to other types of events (e.g., fire). If a network is designed to detect several types of events, sensing modeling for each type is required. Here, we intentionally describe our approach in conceptual terms independent of the concrete method used.

We are targeting sensors like PIR motion sensors, light sensors, etc. However, we do not consider sensors for measure-

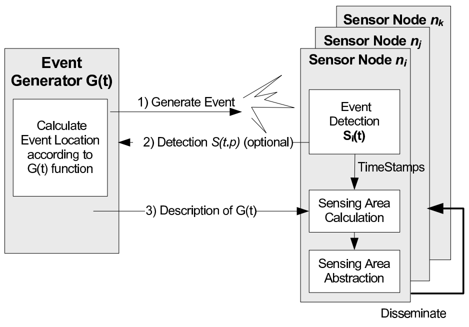

Fig. 1. SAM architecture

ment of temperature, humidity, etc. To identify sensing area, events are generated by real targets. For example, targets (e.g. person or object with mobility) can move around interested area to activate PIR motion sensors in the field. Since the patterns of events are diversified, we describe our approach conceptually independent of the concrete type of events used.

A. Main Idea

The main idea of training-based physical sensing area modeling is to relate the event location to the event detection. Events can be intentionally generated in the space where sensor nodes are deployed. Or, we can collect adequate natural events and information on their locations. We call both types of 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.

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 $k = 2$ 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 $G(t) = p_t = (x_t, y_t)$ where $t \in \{t_1, t_2, ..., t_n\}$.

Figure 1 shows the system architecture of SAM. It consists of two major parts: an event generator G and a set of sensor nodes n_i $(i \in N)$. The event generator G could be a single target or multiple distributed targets that generate a sequence of events $e(t, p)$ with spatiotemporal correlation $G(t) = p(x_t, y_t)$. 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 the case of detection, sensor nodes store the timestamp t locally. By the end of training, a sensor will have computed the location of all the events it detects by inputting the timestamps into $G(t)$. Therefore, a set of timestamps $T_i = \{t^i_1, t^i_2, \ldots, t^i_n\}$ stored in node n_i can be converted to a set of locations $P_i = \{p^i_1, p^i_2, \ldots, p^i_n\}$ within the sensing area. The location set P_i can be directly used to describe the sensing area of node n_i.

III. 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.

Algorithm 1 Regular $G(t)$ Process

```
1: output $P_i$: The sensing area of $n_i$.
2: $T = \emptyset$ //an empty set of timestamps
3: repeat
4:     Event generator $G$ creates $e(t, p)$ at time $t$ and location $p(x, y)$ according to $G(t)$
5:     if node $n_i$ detects event $e(t, p)$, i.e. $S_i(t, p) = 1$ then
6:         it stores the timestamp $t$ into set $T$
7:     end if
8: until $G$ stops generating events
9: Event generator $G$ disseminates the description of $G(t)$ to all nodes
10: Node $n_i$ obtains a set of locations $P_i$ by correlating $G(t)$ with $T_i = \{t^i_1, t^i_2, \ldots, t^i_n\}$
11: $P_i$ is a set of positions $p$ where $S_i(t, p) = 1$
```

A. Regular $G(t)$

To illustrate the basic functionality of an event generator, we start with a simple sensor system where the sensing area of a node is a line segment as shown in Figure 2(a). We shall 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 same interval D. Formally, $G(t) = t \cdot v$, where $t = kD/v$ and $0 \leq k \leq L/D$. For example, in Figure 2(a), 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 timestamps are converted to a set of actual event locations $P_1 = \{t_1v, t_2v, \ldots, t_6v\}$. The sensing area 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 area of sensor n_2 is defined as the line segment that covers $P_2 = \{t_4v, t_5v, t_6v, t_7v\}$. 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 2(a). The regular training can be generalized to the case when the events occur in a plane. Figure 2(b) shows this approach. In this case, training area A is divided into several lines $\alpha_1, \alpha_2, \ldots$, and we can obtain sensing area in a plane in the similar way to the above. In addition to the progressive scanning, the $G(t)$ function of the regular training can generate events with an arbitrary sequence. The detailed operations to identify the sensing area of a single node n_i are described in Algorithm 1.

The advantage of regular training is its simplicity and unidirectional communication. After a node receives the description of $G(t)$, its sensing area can be inferred locally. The detection results $S(t, p)$ do not have to be reported. On the other hand, the event overhead of regular $G(t)$ is a concern, especially when the density of the sensor node is small and the area is large. This motivates us to consider a hierarchical solution.

B. Hierarchical $G(t)$

Hierarchical G(t) is motivated by the observation that the boundary area of a sensing area requires more detail than the

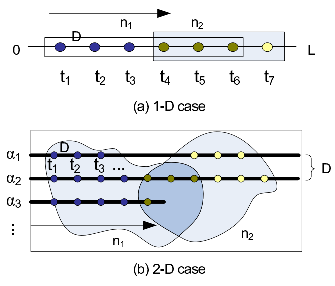

Fig. 2. Regular training

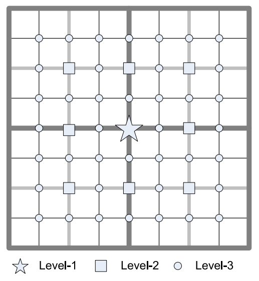

Fig. 3. Hierarchical partition

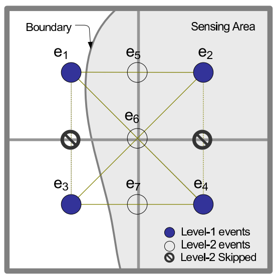

Fig. 4. Level of details

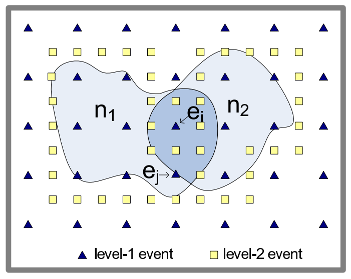

Fig. 5. Hierarchical training

Algorithm 2 Hierarchical G(t) process

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

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, a level-1 event divides the area into four 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. If an event is a level-i event, it is also a level-j event, where $j \geq i$. 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 maximal eight adjacent events). Two adjacent events are said to be a *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 the step by step operation of Hierarchical $G(t)$ in Algorithm 2.

1) A Walkthrough of Hierarchical $G(t)$: We illustrate the main idea how to find 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, not 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 time t_5, t_6, 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, 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 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 shows 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 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.

C. Sensing Area Representation

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 and unnecessary message overhead, especially when the sensing area is large. To address this issue, we can abstract a set of discrete locations (which is estimated to be covered by a sensor) as a polygon by walking across the boundary

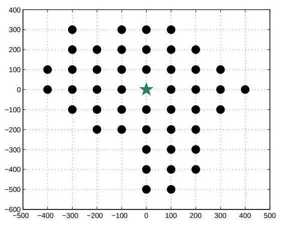
Fig. 6. Coverage without obstacle

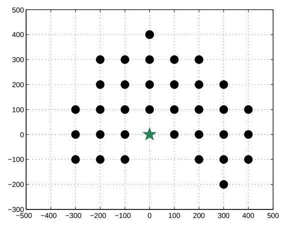
Fig. 7. Coverage with obstacle

TABLE I
SENSING AREA IN OUTDOOR EXPERIMENT

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

points either clockwise or counterclockwise (using the left-hand or right-hand rule as in GPSR [5]). Once the coverage is represented as a polygon by *wrapping*, we can further *simplify the polygon* using the Douglas-Peucker (DP) algorithm [6] in $O(n^2)$ where n is the number of vertices.

IV. OUTDOOR EXPERIMENTS

To evaluate the practicality of our design, we used ExScal XSM motes to obtain empirical results on irregular sensing patterns outdoors. PIR sensors detect movements through changes in infrared radiation, which could be caused by walking persons or moving vehicles. We adopted the regular training approach; however, instead of training the motes using parallel lines as shown in Figure 2b, we used people's natural movement. To map the event time to the event position, we exposed a camera during training. Then, the time an event was detected was compared with the camera capture time on the people's movement, converted to the people's location, and included in the coverage of the detecting sensor node.

Figures 6 and 7 show the sensing area we obtain after training a sensor which is placed (1) in an open area and (2) in an area with a obstacle. A person moved around a sensor sufficiently (10 times straight cross over the area in different directions and positions). The positions belonging to the detected events were associated to the closest grid points which we indicated in the figure. As can be seen in the figure, the sensing area is irregular even without a obstacle. The obstacle affects the sensing area significantly. With the circle model (a disk with radius $4m$), 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 training test, we obtained irregularity and training confidence as shown in Table I. 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.

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

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.

V. EXTENSIVE EVALUATION

Without knowledge of the ground truth of real sensing coverage we can investigate only the characteristics of sensing coverage and the feasibility of our proposed methods for training. In this section, we extend the evaluation of our method by incorporating knowledge of the ground truth.

A. 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), 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 SAM algorithm. The oracle generates a sensing pattern according to the following irregularity model, which is an extension of the DOI model [7].

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

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. $Rand$ is 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.

B. System Implementation and Setup

We designed and implemented a complete version of training which 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 into a 128 by 128 grid 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 (1) with $R_{min} = 10.0, R_{max} = 30.0$ and var = 1.0, 2.0 or 3.0 (default is 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 $i - j + 1$.

C. Evaluation Metrics

We defined (1) **false positive** fp and (2) **false negative** fn error as:

- $fp = \frac{\text{area size included in training but not in reality}}{\text{area size of real sensing coverage}}$
- $fn = \frac{\text{area size not included in training but is in reality}}{\text{area size of real sensing coverage}}$

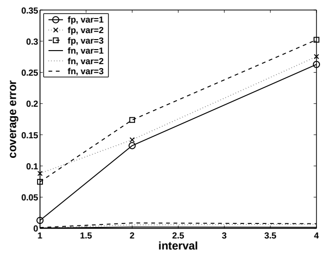

Fig. 8. Errors in regular $G(t)$ with varying interval and irregularity

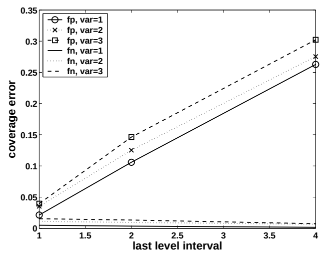

Fig. 9. Errors in hierarchical $G(t)$ with varying interval and irregularity

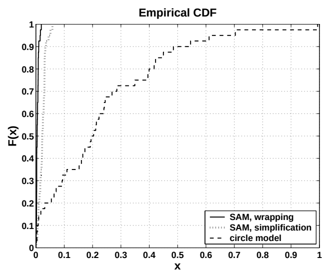

Fig. 10. The CDF fp curves for circular model and two representation methods in SAM model

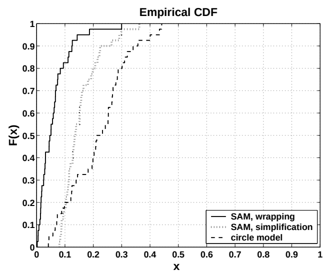

Fig. 11. The CDF fn curves for circular model and two representation methods in SAM model

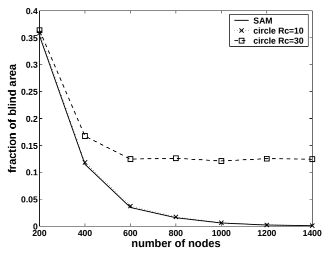

Fig. 12. Fraction of blind area with varying densities

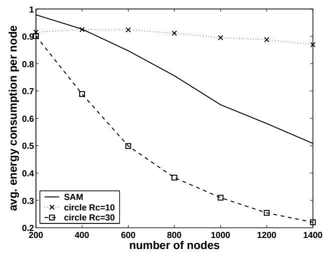

Fig. 13. Avg. energy consumed with varying densities

D. fp and fn of Sensing Coverage

Coverage error increases under the following two conditions (i) the irregularity of sensing area increases, or (ii) the training interval 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 training interval, but different last-level training intervals. Figure 8 shows that with a small training interval, we can achieve very precise coverage modeling. fp is almost 0% and fn is at 1% to 8%. The coverage error in Figure 8 for a certain fixed interval in the regular training is very similar to the coverage error in Figure 9 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.

E. Distribution of fp and fn

We compared the CDF curves of fp and fn under three settings: circular model, SAM model with polygon wrapping, and SAM model with polygon simplification (described in Section III-C). In the circular model, the sensing coverage is assumed to be a disk at the center of the sensor location with radius $\frac{R_{min}+R_{max}}{2} = 20$. Since the simplification approach uses fewer vertices to describe the area, it is less accurate than the wrapping. From Figure 10 and 11, we can clearly see that SAM significantly outperform the circular model in terms of fp and fn rates.

VI. Application Improvements

In evaluation, we apply full coverage scheduling [8] based on individual sensor coverage by a circle model and by the SAM training model. The design goal of full coverage scheduling is to cover every physical point within an area with minimal energy consumption. The fraction of blind area and energy consumption are two key metrics for coverage applications. Figure 12 shows the fraction of blind area when different node densities are provided for a given deployment area. As we increased the number of nodes from 200 to 1400, the blind area by coverage scheduling in SAM significantly decreases. On the other hand, with optimistic circular model (a disk with radius $R_c = 30$), the percentage of blind area stays at about 15%, despite the fact that over 1400 nodes have been deployed into the area. Figure 13 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 SAM has accurate sensing area information with smaller energy consumption.

VII. Conclusion

This paper intends to draw attention to the sensing irregularity issue known but largely ignored by many designers. We contribute to this area by designing two training-based methods that accurately identify the sensing patterns. Our design has been fully implemented and evaluated by outdoor experiment as well as by indoor emulation. Also importantly, the impacts of sensing irregularity on typical application are identified and the improvements by SAM are shown as well. We hope this work motivates our community to seriously consider the reality issues existed in the sensor networks.

References

[1] J. Liu, J. Reich, and F. Zhao, "Collaborative In-Network Processing for Target Tracking," *J. on Applied Signal Processing*, March 2003.

[2] R. Szewczyk, A. Mainwaring, J. Anderson, and D. Culler, "An Analysis of a Large Scale Habit Monitoring Application," in *SenSys'04*, 2004.

[3] 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*, 2004.

[4] 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'05*, November 2005.

[5] B. Karp and H. T. Kung, "Greedy Perimeter Stateless Routing for Wireless Networks," in *MobiCom'00*, 2000.

[6] 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), 112-122*, 1973.

[7] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "Range-Free Localization Schemes in Large-Scale Sensor Networks," in *MOBICOM'03*, September 2003.

[8] T. Yan, T. He, and J. A. Stankovic, "Differentiated Surveillance Service for Sensor Networks," in *SenSys'03*, November 2003.