Localized Outlying and Boundary Data Detection in Sensor Networks

Weili Wu, *Member*, *IEEE*, Xiuzhen Cheng, *Member*, *IEEE*, Min Ding, *Student Member*, *IEEE*, Kai Xing, *Student Member*, *IEEE*, Fang Liu, *Student Member*, *IEEE*, and Ping Deng, *Student Member*, *IEEE*

Abstract—This paper targets the identification of outlying sensors (that is, outlying reading sensors) and the detection of the reach of events in sensor networks. Typical applications include the detection of the transportation front line of some vegetation or animalcule's growth over a certain geographical region. We propose and analyze two novel algorithms for outlying sensor identification and event boundary detection. These algorithms are purely localized and, thus, scale well to large sensor networks. Their computational overhead is low, since only simple numerical operations are involved. Simulation results indicate that these algorithms can clearly detect the event boundary and can identify outlying sensors with a high accuracy and a low false alarm rate when as many as 20 percent sensors report outlying readings. Our work is exploratory in that the proposed algorithms can accept any kind of scalar values as inputs—a dramatic improvement over existing work, which takes only 0/1 decision predicates. Therefore, our algorithms are generic. They can be applied as long as "events" can be modeled by numerical numbers. Though designed for sensor networks, our algorithms can be applied to the outlier detection and regional data analysis in spatial data mining.

Index Terms—Sensor networks, event boundary detection, outlying sensor identification, ROC curve analysis.

1 Introduction

In ecological studies, sensor networks can be deployed to monitor the *invasive species spread*. This represents a class of sensor network applications in which events (phenomena) span a relatively large geographic region. In this paper, we consider the detection of the event boundary. This is an important task in sensor networks for many reasons. For example, the reach of the special vegetation or an animalcule's growth provides ecologists with the most important information.

Event boundary detection is a challenging problem. As understood by researchers, sensor networks suffer from a very limited resource provisioning. Further, sensors are error prone due to low cost; thus, they are usually densely deployed to compensate for each other. Therefore, in designing algorithms for boundary detection, we face the problem of efficiently processing a large volume of data containing redundant and spurious information. Moreover, we need to disambiguate an outlying reading and a reading that signals an event, since they are indistinguishable in nature.

Our objective is to design localized algorithms to identify outliers (outlying sensors) and event sensors at an event boundary. As reported earlier, an outlying reading and a reading signaling an event in a sensor network may not be distinguishable. However, outlying readings are geographically independent, whereas normal sensors observing the same phenomenon are spatially correlated [11]. These observations constitute the base for our algorithm design.

We first propose an algorithm to identify outlying sensors. These sensors may report outlying values due to hardware defects or environmental variations. The basic idea of outlying sensor detection is given as follows: Each sensor first computes the difference between its reading and the median reading from the neighboring readings. Each sensor then collects all differences from its neighborhood and standardizes them. A sensor is an outlier if the absolute value of its standardized difference is sufficiently large.

We then propose an algorithm for event boundary detection. This algorithm is based on the outlying sensor detection algorithm and the following simple observation. For an event sensor, there often exist two regions, with each containing the sensor, such that the absolute value of the difference between the reading of the sensor and the median reading from all other sensors in one region is much larger than that in another region.

Our algorithms require threshold values for comparison. We propose an adaptive threshold determination mechanism based on the receiver-operating characteristic (ROC) curve analysis. The performances of these algorithms are verified by extensive simulation study.

Special features of our algorithms include the following:

1. The input can be any numeric value. This is significantly different from the existing work [6], [11] based on 0/1 decision predicates, where 1

- *W. Wu and P. Deng are with the Department of Computer Science, University of Texas at Dallas, Mail Station EC31, Richardson, TX 75083. E-mail: {weiliwu, pxd010100}@utdallas.edu.*
- *X. Cheng, M. Ding, K. Xing, and F. Liu are with the Department of Computer Science, The George Washington University, 801 22nd Street, NW, Suite 704, Washington, DC 20052. E-mail: {cheng, minding, kaix, fliu}@gwu.edu.*

Manuscript received 16 Feb. 2006; revised 5 July 2006; accepted 15 Jan. 2007; published online 29 Jan. 2007.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0078-0206.
Digital Object Identifier no. 10.1109/TKDE.2007.1062.

indicates the occurrence of some phenomenon, and 0 indicates a normal status.
2. The computation overhead is low, which involves only simple algebraic operations.
3. The communication overhead is low, since sensor readings are disseminated to the neighborhood only.
4. Event boundaries can be accurately identified even when many sensors report outlying measurements. In other words, outliers and boundary sensors can be clearly differentiated.

This paper is organized as follows: In Section 2, we summarize the adopted network model and major related work. The two localized algorithms for outlier identification and event boundary detection are proposed in Sections 3 and 4, respectively. Performance metrics and analysis are outlined in Section 5. Our simulation results are reported in Section 6. We conclude our paper in Section 7 with a future research discussion.

2 Network Model and Related Work

2.1 Network Model

Throughout this paper, we assume that N sensors are uniformly distributed in the network area, with a base station residing in the boundary. The network region is a $b \times b$ squared field located in the two-dimensional euclidean plane $\mathcal{R}^2$. A sensor's reading is *outlying* if it deviates significantly from other readings of neighboring sensors [1]. Sensors with outlying readings are called *outliers* or *outlying sensors*, whereas sensors with normal readings are called *normal sensors*. In this paper, the ith sensor S_i and its location will be used interchangeably. We use S to denote the set of all the sensors in the field, and R denotes the radio range of the sensors. Let x_i denote the reading of the sensor S_i. Instead of a 0-1 binary variable, x_i is assumed to represent the actual reading of a factor or variable such as temperature, light, sound, the number of occurrences of some phenomenon, and so on. Therefore, x_i can be continuous or discrete.

Informally, an event can be defined in terms of sensor readings. An *event*, denoted by $\mathcal{E}$, is a subset of $\mathcal{R}^2$ such that the readings of the sensors in $\mathcal{E}$ are significantly different from those of sensors that are not in $\mathcal{E}$. A sensor detecting some event is called an *event sensor*. An outlying sensor can be viewed as a special event that contains only one point, that is, the sensor itself. A point $\mathbf{x} \in \mathcal{R}^2$ is said to be *on the boundary of* $\mathcal{E}$ if and only if each closed disk centered at $\mathbf{x}$ contains both points in $\mathcal{E}$ and points that are not in $\mathcal{E}$. The *boundary* of the event $\mathcal{E}$, denoted by $\mathrm{B}(\mathcal{E})$, is the collection of all the points on the boundary of $\mathcal{E}$.

We assume that each sensor can locate its physical position through either GPS or GPS-less techniques such as [5], [14], [20]. Note that in this paper, we focus on the detection of outliers and event boundary sensors; thus, report generation and delivery to the base station will not be considered. Further, we assume that there exists a media access control (MAC) layer protocol to coordinate neighboring broadcastings such that no collision occurs.

2.2 Related Work

Spatial outlier detection and regional data analysis have been extensively studied in spatial data mining research [12], [15], [16]. In this section, we briefly survey related results in sensor network research.

As we have noted earlier, when a remarkable change in the readings of sensors is detected, an outlier or some event must have occurred. This observation is explored in [4], [13], [17] for 0/1 decision predicate computation. The related algorithms require only the most recent readings (within a sliding window) of individual sensors. No collaboration among neighboring sensors is exploited. In [4], the "change points" of the time series are statistically computed. The detector proposed in [13] computes a running average and compares it with a threshold, which can be adjusted by a false alarm rate. In [17], *kernel density estimators* are designed to check whether the number of outlying readings is beyond an application-specific threshold. Note that none of these works can disambiguate outlying sensors and real event sensors, since only observations from individual sensors are studied. By exploring the correlation among neighboring sensors, our algorithms can discern outlying sensors from event sensors and compute the boundary of the event region.

For outlying or misbehaving node identification, one solution is to seek the help of the base station [18], [19]. Staddon et al. [18] propose tracing failed nodes in sensor networks at a base station, assuming that all sensor measurements will be directed to the base station along a routing tree. In this work, the base station that has a global view of the network topology can identify failed nodes through route update messages. In [19], base stations launch marked packets to probe sensors and rely on their responses to identify and isolate insecure locations. Our algorithm is more versatile. It is purely localized, which can save a great amount of communication overhead and therefore help elongate the network lifetime.

To the best of our knowledge, the only work that targets localized event boundary detection in sensor networks is the one in [6], in which three proposed schemes take as inputs the 0/1 decision predicates from neighboring sensors. The work for event region detection in [11] also takes only 0/1 decision predicates. This is a major drawback as compared to our work, which does not impose such restrictions. Further, the threshold in our algorithms can be determined based on the ROC curve analysis, which is computationally efficient as compared to those in [6] and [11]. Using 0/1 decision predicates for boundary computation may have the following disadvantages: 1) The 0/1 decision predicates are the results of comparing current sensor readings with a threshold. If the threshold is a global cut based on a priori information, then the 0/1 predicates will miss the spatial information on deployed sensors. 2) The 0/1 decision predicates are the preprocessed results of the actual measured data. Detection over binary predicates represents the second-round approximation. 3) The 0/1 predicates may not be correct due to faulty sensors. Intuitively, algorithms based on original sensor readings or measurements should be more precise and more robust.

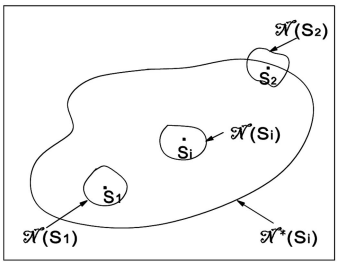


Fig. 1. An $\mathcal{N}^*$ neighborhood of sensor S_i and $\mathcal{N}$ neighborhoods of sensors inside $\mathcal{N}^*(S_i)$. Each $\mathcal{N}$ neighborhood is used to compute d_i, whereas the $\mathcal{N}^*(S_i)$ is used to compare the d_i's.

3 Localized Outlying Sensor Detection

In this section, we describe our algorithm for detecting sensors whose readings (measurements) deviate considerably from their neighbors.

3.1 Derivation of Detection Procedure

The procedure of locating outliers in sensor networks could be formalized statistically as follows: Consider how we can compare the reading at S_i with those of its neighbors. Let $\mathcal{N}(S_i)$ denote a bounded closed set of $\mathcal{R}^2$, which contains the sensor S_i and additional k sensors $S_{i1}, S_{i2}, \cdots, S_{ik}$. The set $\mathcal{N}(S_i)$ represents a closed neighborhood of the sensor S_i. An example of $\mathcal{N}(S_i)$ is the closed disk centered at S_i, with the radius R. Let $x_1^{(i)}, x_2^{(i)}, \cdots, x_k^{(i)}$ denote the measurement at $S_{i1}, S_{i2}, \cdots, S_{ik}$, respectively. A comparison between x_i and $\{x_1^{(i)}, x_2^{(i)}, \cdots, x_k^{(i)}\}$ could be done by checking the difference between x_i and the "center" of $\{x_1^{(i)}, x_2^{(i)}, \cdots, x_k^{(i)}\}$. Clearly, such a difference is

$$d_i = x_i - \mathrm{med}_i, \tag{1}$$

where med_i denotes the median of the set $\{x_1^{(i)}, x_2^{(i)}, \cdots, x_k^{(i)}\}$. We note that the med_i in (1) should not be replaced by the mean $(x_1^{(i)} + x_2^{(i)} + \cdots + x_k^{(i)})/k$ of the set $\{x_1^{(i)}, x_2^{(i)}, \cdots, x_k^{(i)}\}$. This is because the sample mean cannot represent the "center" of a sample well when some values of the sample are extreme. However, median is a robust estimator of the "center" of a sample. If d_i is large or large but negative, then it is very likely that S_i is an outlier. Now, we start quantifying the degree of extremeness of d_i. To do this, we need the differences d that are associated with sensors that are near S_i and computed via (1).

Consider another bounded closed set $\mathcal{N}^*(S_i) \subset \mathcal{R}^2$, which contains S_i and additional $n-1$ sensors. This set $\mathcal{N}^*(S_i)$ also represents a neighborhood of S_i. Among many choices of $\mathcal{N}^*(S_i)$, one could select $\mathcal{N}^*(S_i) = \mathcal{N}(S_i)$. We denote the n sensors in $\mathcal{N}^*(S_i)$ by $S_1, \cdots, S_i, \cdots, S_n$ (see Fig. 1 for an illustration of $\mathcal{N}$ and $\mathcal{N}^*$). According to (1), sensors in $\mathcal{N}^*(S_i)$ yield $d_1, \cdots, d_i, \cdots, d_n$. Now, if d_i is extreme in $D = \{d_1, \cdots, d_i, \cdots, d_n\}$, then S_i will be treated as an outlying sensor. The decision can be made vigorously by using the following procedure. Let $\hat{\mu}$ and $\hat{\sigma}$ denote, respectively, the sample mean and sample standard deviation of the set D, that is,

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} d_i$$

$$\hat{\sigma} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(d_i - \hat{\mu})^2}.$$

Standardize the data set D to obtain $\{y_1, \cdots, y_i, \cdots, y_n\}$, where

$$y_1 = \frac{d_1 - \hat{\mu}}{\hat{\sigma}}, \cdots, y_i = \frac{d_i - \hat{\mu}}{\hat{\sigma}}, \cdots, y_n = \frac{d_n - \hat{\mu}}{\hat{\sigma}}. \tag{2}$$

Decision. *If $|y_i| \geq \theta$, then treat S_i as an outlying sensor. Here, $\theta(> 1)$ is a preselected number.*

We now start justifying the above decision-making procedure under certain assumptions. For this purpose, we first need some result of the median. Given $\mathcal{N}(S_i)$, assume that $x_1^{(i)}, x_2^{(i)}, \cdots, x_k^{(i)}$ form a sample from a population having a continuous distribution function F. Let $x_1^{(i)}, x_2^{(i)}, \cdots, x_k^{(i)}$ be rearranged in the order from least to greatest and let the ordered values be $x_{(1)}^{(i)}, x_{(2)}^{(i)}, \cdots, x_{(k)}^{(i)}$, where $x_{(1)}^{(i)} \leq x_{(2)}^{(i)} \leq \cdots \leq x_{(k)}^{(i)}$. Then, (1) can be rewritten as

$$\mathrm{med}_i = \begin{cases} x_{((k+1)/2)}^{(i)} & \text{if } k \text{ is odd} \\ (x_{(k/2)}^{(i)} + x_{(k/2+1)}^{(i)})/2 & \text{if } k \text{ is even.} \end{cases} \tag{3}$$

Assuming that the median of the distribution F is $\tilde{\mu}$, and $F(\tilde{\mu}) = 0.5$ has a unique solution, we have the following:

Proposition. *As $k \to \infty$, med_i converges in probability to $\tilde{\mu}$.*

To prove this, we first note the following special case of Theorem 9.6.5 in [21]: If kp_k is a positive integer such that $p_k = 0.5 + O(1/k)$, then, as $k \to \infty$, $x_{(kp_k)}^{(i)}$ converges in probability to $\tilde{\mu}$. For any real number a, let $\lfloor a \rfloor$ denote the largest integer that is less than or equal to a and let (a) denote the difference $a - \lfloor a \rfloor$. Then, $0 \leq (a) < 1$. Set $p_{1k} = \frac{\lfloor 0.5k \rfloor + 1}{k}$. Then,

$$p_{1k} = \frac{0.5k - (0.5k) + 1}{k} = 0.5 + O\left(\frac{1}{k}\right).$$

Let $p_{2k} = p_{1k} - 1/k$. Then, $p_{2k} = 0.5 + O(1/k)$. Therefore, both $x_{(kp_{1k})}^{(i)}$ and $x_{(kp_{2k})}^{(i)}$ converge in probability to $\tilde{\mu}$. Now, the proposition follows from the observation that (3) is equivalent to the following:

$$\mathrm{med}_i = \begin{cases} x_{(kp_{1k})}^{(i)} & \text{if } k \text{ is odd} \\ (x_{(kp_{1k})}^{(i)} + x_{(kp_{2k})}^{(i)})/2 & \text{if } k \text{ is even.} \end{cases}$$

The above property of median is established for a quite general class of F. Deeper results of median are also available. For example, an asymptotic normal distribution of median can be obtained under some general conditions [3].

Now, consider the following simple scenario, where 1) readings of sensors in $\mathcal{N}^*(S_i)$, that is, $x_1, \cdots, x_n$, are independent, 2) for each sensor S_j in $\mathcal{N}^*(S_i)$, the readings from the sensors in $\mathcal{N}(S_j)$ form a sample of a normal distribution, and 3) all the variances of the above mentioned distributions are equal. Since the median is equal to the

mean for any normal distribution, it follows from the proposition and from 1)-3) that, as k becomes large, the sequence $d_1, \cdots, d_n$ forms approximately a sample from a normal distribution with mean equal to 0. This implies that as k is large, the sequence from standardization, that is, $y_1, \cdots, y_n$, can be treated as a sample from a standard normal population $N(0,1)$. When x_i is particularly large or small, compared with all of the other x values, d_i will deviate markedly from all the other d values. Consequently, $|y_i|$ will be large, which implies that y_i will fall into the tail region of the density of the standard normal population. However, if $|y_i|$ is large, the probability of obtaining this observation y_i is small and, thus, S_i should be treated as outlying. When $\theta = 2$, the probability of observing a y_i with $|y_i| \geq 2$ is about 5 percent.

3.2 Algorithm

Let $\mathcal{C}_1$ denote the set of sensors with $|y_i| \geq \theta$. The set $\mathcal{C}_1$ is viewed as a set of sensors that are claimed as outliers by the above procedure. The procedure in Section 3.1 can be summarized into the following algorithm:

Algorithm 1

1. Construct $\{\mathcal{N}\}$ and $\{\mathcal{N}^*\}$. For each sensor S_i, perform the following steps.
2. Use $\{\mathcal{N}(S_i)\}$ and (1) to compute d_i for sensor S_i.
3. Use $\{\mathcal{N}^*(S_i)\}$ and (2) to compute y_i for sensor S_i.
4. If $|y_i| \geq \theta$, where $\theta > 1$ is predetermined, assign S_i to $\mathcal{C}_1$. Otherwise, S_i is treated as a normal sensor.

Clearly, $|\mathcal{C}_1|$, which is the size of $\mathcal{C}_1$, depends on θ. Assuming that the y values in (2) constitute a sample of a standard normal distribution and the decisions are made independently, then, if θ is chosen so that the right-tail area of the density of $N(0,1)$ is α, $|\mathcal{C}_1|$ will be about $\alpha \times N$.

In practice, a sensor becomes outlying if 1) data measurement or data collection makes errors, 2) some variability in the area surrounding the sensor has changed significantly, or 3) the inherent function of the sensor is abnormal. In any of the three cases, readings from outlying sensors do not reflect reality, so they can be discarded before further analysis on sensor data. However, outlying readings may contain valuable information related to events and provide help in detecting the events. For this reason, issues concerning event region detection will be addressed in the presence of data from outlying sensors.

4 Localized Event Boundary Detection

In this section, we describe our procedure for localized event region detection. To detect an event region, it suffices to detect the sensor nodes near or on the boundary of the event. We note that $\mathcal{C}_1$ may contain some normal sensors close to the event boundary. However, Algorithm 1 usually does not effectively detect sensors close to the boundary of the event. To illustrate this point, let us consider the simple situation where the event lies to one side of a straight line. Suppose that readings of sensors in the event (region) $\mathcal{E}$ form a sample from a normal distribution $N(\mu_1, \sigma^2)$, and sensor readings outside $\mathcal{E}$ form another sample from $N(\mu_2, \sigma^2)$, where $\mu_1 \neq \mu_2$, and σ is small as compared to $|\mu_1 - \mu_2|$. $\{\mathcal{N}\}$ and $\{\mathcal{N}^*\}$ are constructed using closed disks.

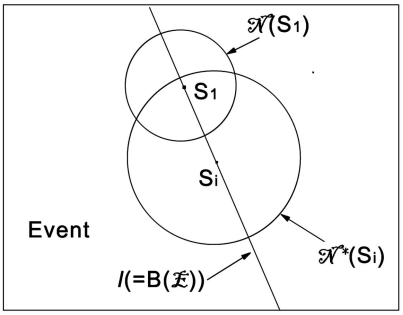


Fig. 2. Event $\mathcal{E}$ is the union of line l and the portion on the left-hand side of l. S_i is a sensor located on $\mathrm{B}(\mathcal{E})$, and S_1 is a sensor inside $\mathcal{N}^*(S_i)$. Both $\mathcal{N}^*(S_i)$ and $\mathcal{N}(S_1)$ are closed disks.

Consider a sensor S_i that is close to the event boundary. Below, we will show that even under some quite favorable setting, such a sensor S_i may not be detected, that is, assigned to $\mathcal{C}_1$ by Algorithm 1.

Assume that readings of sensors in a neighborhood of S_i are within 2σ distance from the means of their corresponding normal distributions. Take each $\mathcal{N}$ neighborhood of sensors in $\mathcal{N}^*(S_i)$ to be sufficiently large. Due to uniformity of the deployment of sensors, calculation based on (1) shows that each d follows $N(0, \sigma^2)$ approximately. For example, at sensor S_1 shown in Fig. 2, d_1 follows approximately $N(0, \sigma^2)$. The reasoning is explained as follows: Let R_1 denote the portion of $\mathcal{N}(S_1)$ that lies on the right-hand side of the event boundary and R_2 denote the remaining portion of $\mathcal{N}(S_1)$. Then, the area of R_1 is larger than that of R_2. Since the sensors are uniformly distributed, the expected number of sensors in R_1 is larger than the expected number of sensors in R_2. Then, med_1 will be obtained using the sensor readings from R_1. When σ is small compared to $|\mu_1 - \mu_2|$, med_1 is about μ_2 so that d_1, which is about $x_1 - \mu_2$, follows approximately $N(0, \sigma^2)$. Furthermore, it is seen that $y_1, \cdots, y_n$ from (2) form approximately a sample of $N(0,1)$, where each member of the sample is within distance 2 of the mean 0. Therefore, $\mathcal{C}_1$ from Algorithm 1 will not pick up this sensor S_i if $\theta = 2$.

So that sensors near and on $B(\mathcal{E})$ can be detected efficiently, the procedure described in Algorithm 1 should be modified. As motivated above, to detect a sensor S_i close to the boundary, we should select a special neighborhood $\mathcal{NN}(S_i)$ such that d_i, compared with d values from surrounding neighborhoods, is as extreme as possible. There are many options in doing this. Here, we describe two of them: random bisection and random trisection.

4.1 Random Bisection

Consider an S_i from the set $S - \mathcal{C}_1$. Place a closed disk centered at S_i. Randomly draw a line through S_i, dividing the disk into halves. Calculate d_i in each half. Use $\mathcal{NN}(S_i)$ to denote the half disk yielding the largest $|d_i|$. For an illustration, see Fig. 3. In the figure, the line randomly chosen meets the boundary of the disk at points P_1 and P_2, and the boundary of the event meets the boundary of the disk at points A and B. Due to uniformity of sensor deployment, we see that $|d_i|$, from the half disk containing P_1, P_2, B, and S_i, is the largest and, hence, this half will be used as $\mathcal{NN}(S_i)$. After $\mathcal{NN}(S_i)$ is found, the resulting d_i will be used to replace the old d_i, keeping unchanged all of

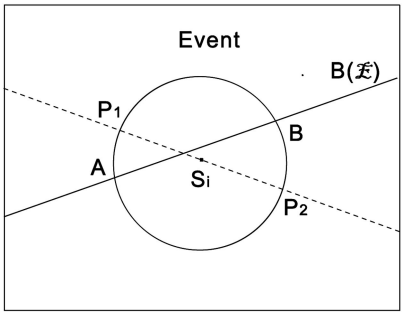


Fig. 3. Illustration of random bisection. $\mathcal{NN}(S_i)$ is the half disk containing P_1, S_i, P_2, and B.

the other d values from $\mathcal{N}^*(S_i)$. Then, perform the calculation in (2) and make a decision on S_i. We note that for random bisection, as well as random trisection (introduced next), lines are drawn randomly to form sectors simply due to the fact that the location and the shape of an event boundary are usually unknown a priori.

4.2 Random Trisection

Consider a closed disk centered at $S_i \in S - \mathcal{C}_1$. Randomly divide the disk into three sectors with an equal area. Number the sectors as i, ii, and iii, as shown in Fig. 4. Form a union by using any two sectors and calculate d_i in each union ($\text{total} = 3$). The union resulting in the largest $|d_i|$ is $\mathcal{NN}(S_i)$. It is easy to see in Fig. 4 that $\mathcal{NN}(S_i)$ is the union of sectors i and iii. The d_i with the largest $|d_i|$ will replace the previous d_i, keeping all of the other d values unchanged from $\mathcal{N}^*(S_i)$, and, subsequently, a decision will be made on S_i.

4.3 Event Boundary Determination

There are two options for the decision on S_i, based on $\mathcal{NN}(S_i)$. If $|y_i| < \theta$, S_i would be treated as a normal sensor. If $|y_i| \geq \theta$, the sensor S_i can be close to or far away from the boundary $\mathrm{B}(\mathcal{E})$. Let $\mathcal{C}_2$ denote the set of all the sensors with $|y_i| \geq \theta$. The set $\mathcal{C}_2$ is expected to contain enough sensors close to the event boundary (see Fig. 5c). In general, $\mathcal{C}_2$ catches more sensors near the event boundary than $\mathcal{C}_1$ does. Now, we discuss how we can combine $\mathcal{C}_1$ and $\mathcal{C}_2$ to infer the event boundary.

As seen in the derivation of $\mathcal{C}_1$ and $\mathcal{C}_2$, the set $\mathcal{C}_1$ is expected to contain outlying sensors, and $\mathcal{C}_2$ is expected to contain sensors close to the event boundary. However, in general, $\mathcal{C}_1$ also contains some sensors near the boundary, which are not outlying, and $\mathcal{C}_2$ contains some sensors that are not close to the boundary. We now present a method to combine $\mathcal{C}_1$ and $\mathcal{C}_2$ to form a set of sensors that can be used to infer the outline of the event boundary. Consider how we can select sensors from the union $\mathcal{C}_1 \cup \mathcal{C}_2$ to approximate the boundary. For a sensor $S_i \in \mathcal{C}_1 \cup \mathcal{C}_2$, draw a closed disk $D(S_i; c)$ with radius c, centered at S_i. The expected number of sensors falling into the disk is $m = \frac{\pi c^2 N}{b^2}$. Since sensor readings are usually correlated, and $\mathcal{C}_2$ mainly contributes to the set of sensors near the event boundary, S_i is expected to be close to the boundary if $D(S_i; c)$ contains at least one sensor from $\mathcal{C}_2$, which is different from S_i. For any positive integer m, let $\mathcal{C}_3(m)$ denote the subset of $\mathcal{C}_1 \cup \mathcal{C}_2$ such that for each $S_i \in \mathcal{C}_3(m)$, the disk $D\left(S_i; \sqrt{\frac{mb^2}{\pi N}}\right)$ contains at least one sensor from $\mathcal{C}_2$ that is different from S_i. The set $\mathcal{C}_3(m)$ will serve as a set of sensors used to infer the event boundary (see Fig. 5d). For convenience, sometimes, we will write $\mathcal{C}_3$ as $\mathcal{C}_3(m)$.

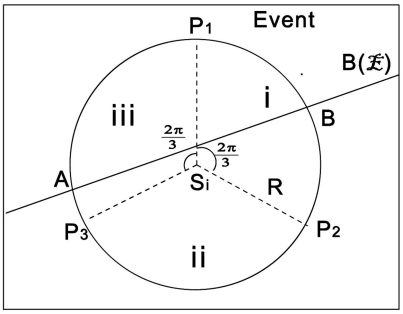


Fig. 4. Illustration of random trisection. Sectors $P_1S_iP_2$, $P_2S_iP_3$, and $P_3S_iP_1$ are numbered as i, ii, and iii, respectively. Each sector contains an angle equal to $2\pi/3$. $\mathcal{NN}(S_i)$ is the union of sectors i and iii.

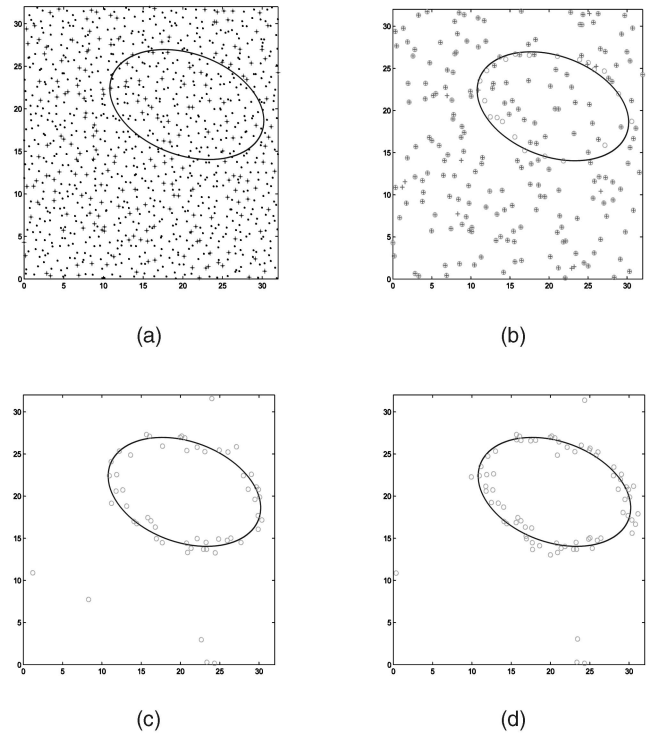


Fig. 5. Illustration of $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$. Data in (a) are obtained from one run of the experiment, leading to panels in the third row of Fig. 12. The interior of the ellipse is the event region. A sensor becomes outlying, with probability $p = 0.2$. A $\cdot$ represents a sensor and a $+$ represents an outlying sensor. A $\circ$ represents a node in (b) $\mathcal{C}_1$, (c) $\mathcal{C}_2$, and (d) $\mathcal{C}_3$, respectively.

Now, we summarize the above procedure of finding $\mathcal{C}_2$ and $\mathcal{C}_3$ into the following algorithm:

Algorithm 2

1. Construct $\{\mathcal{N}\}$ and $\{\mathcal{N}^*\}$. Apply Algorithm 1 to produce the set $\mathcal{C}_1$ ($\theta = \theta_1$).

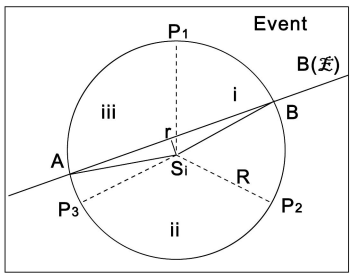


Fig. 6. The event boundary intersects the disk $D(S_i;R)$ in two sectors: i and iii. A and B are two intersection points between the event boundary and the boundary of the disk. r is the distance from sensor S_i to the event boundary.

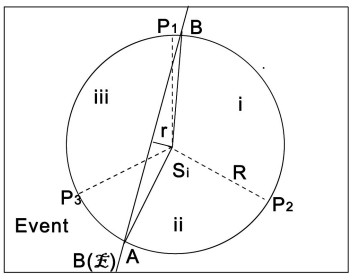


Fig. 7. The event boundary intersects the disk $D(S_i;R)$ in three sectors: i, ii, and iii. A and B are two intersection points between the event boundary and the boundary of the disk. r is the distance from sensor S_i to the event boundary.

2. For each sensor $S_i \in S - \mathcal{C}_1$, perform the following steps. Obtain $\mathcal{NN}(S_i)$ and update d_i from Step 1 to the new d_i from $\mathcal{NN}(S_i)$, keeping unchanged all of the other d values from $\mathcal{N}^*(S_i)$ obtained in Step 1. Use (2) to recompute y_i. If $|y_i| \geq \theta$, assign S_i to set $\mathcal{C}_2$ $(\theta = \theta_2)$; otherwise, treat S_i as a normal sensor.
3. Obtain $\mathcal{C}_3(m)$, where m is a predetermined positive integer.

We stress on the following points on the use of Algorithm 2. First, the updated d_i in Step 2 is only needed when making a decision on sensor S_i. Once such a decision is made, this new d_i will have to be changed back to the original one obtained in Step 1. Second, assuming that the y values in (2) constitute a sample of a standard normal distribution and the decisions are made independently. If θ_1 and θ_2 are such that the right-tail areas of the density of $N(0,1)$ are α_1 and α_2, respectively, the size of $\mathcal{C}_2$ is about $(1-\alpha_1) \times \alpha_2 \times N$. Third, unlike Algorithm 1, which utilizes the topological information of sensor locations to find $\mathcal{C}_1$, Algorithm 2 uses the geographical information of locations to locate $\mathcal{C}_2$ and $\mathcal{C}_3$.

To conclude this section, we make the following note on the computational complexity of Algorithms 1 and 2. Assume $\mathcal{N}^*(S_i) = \mathcal{N}(S_i)$ for each sensor S_i, which is usually the case in a densely deployed sensor network. Let ρ denote the average number of sensors in a typical neighborhood $\mathcal{N}$, which is also the average number of neighboring sensors in the given network. Then, it can be easily shown that both Algorithms 1 and 2 have a computational complexity of $O(\rho\, log\rho)$. Since each sensor broadcasts a constant number of messages, the message complexity of both algorithms is $O(\rho)$.

5 Performance Evaluation

Evaluation of the proposed algorithms includes two tasks: evaluating $\mathcal{C}_1$ and evaluating $\mathcal{C}_3$. In this section, we first define metrics to evaluate $\mathcal{C}_1$. Then, we examine what type of sensors could be detected as being in $\mathcal{C}_2$ by using Algorithm 2. The finding is then used to define metrics to evaluate the performance of $\mathcal{C}_3$.

5.1 Evaluation of $\mathcal{C}_1$

To evaluate the performance of $\mathcal{C}_1$, we compute the *detection accuracy* $a(\mathcal{C}_1)$, defined to be the ratio of the number of outlying sensors detected to the total number of outlying sensors, and the *false alarm rate* $e(\mathcal{C}_1)$, defined to be the ratio of the number of normal sensors that are claimed as outlying to the total number of normal sensors. Let $\mathcal{O}$ denote the set of outliers in the field, then

$$a(\mathcal{C}_1) = \frac{|\mathcal{C}_1 \cap \mathcal{O}|}{|\mathcal{O}|}, \quad e(\mathcal{C}_1) = \frac{|\mathcal{C}_1 - \mathcal{O}|}{N - |\mathcal{O}|}. \tag{4}$$

If $a(\mathcal{C}_1)$ is high, and $e(\mathcal{C}_1)$ is low, Algorithm 1 has a good performance.

5.2 When Could Sensors Be Assigned to $\mathcal{C}_2$?

Here, we present a brief examination on what kind of sensors have the potential to be detected by Algorithm 2 as belonging to $\mathcal{C}_2$. Let r denote the distance from the sensor S_i to the event boundary. We now informally show that if r is larger than $R/2$, then the chance that S_i will not be detected by Algorithm 2 is high. We first consider the case where the random trisection method is used in obtaining $\{\mathcal{NN}\}$.

Let $D(S_i;R)$ denote the closed disk of radius R centered at S_i. Without loss of generality, we may assume that the portion of the boundary falling into $D(S_i;R)$ is a line segment. Clearly, if $\mathrm{B}(\mathcal{E})$ does not intersect $D(S_i;R)$ or intersects $D(S_i;R)$ only in one sector, it is very likely that d_i from the resulting neighborhood $\mathcal{NN}(S_i)$ will not become extreme among d values from $\mathcal{N}^*(S_i)$ so that S_i may not be detected as a sensor in $\mathcal{C}_2$. Therefore, we only need to consider two cases, shown in Figs. 6 and 7, where $\mathrm{B}(\mathcal{E})$ intersects $D(S_i;R)$ in at least two sectors.

Consider Fig. 6, where $\mathrm{B}(\mathcal{E})$ intersects $D(S_i;R)$ in two sectors. Clearly, $\mathcal{NN}(S_i)$ should be the union of sectors i and iii. The event boundary cuts $\mathcal{NN}(S_i)$ into two parts. The part occupied by the event, that is, the part containing P_1, has the following area:

$$A_1 = \frac{\pi R^2}{2\pi}\omega - \frac{1}{2}R^2 \sin\omega,$$

where $\omega \in (0,\pi]$ is the value of $\angle AS_iB$. Consequently, the area of the other part is given as follows:

$$A_2 = \frac{\pi R^2}{2\pi}\left(\frac{4\pi}{3}\right) - A_1.$$

So that d_i from $\mathcal{NN}(S_i)$ becomes extreme, we require $A_1 \geq A_2$, which implies that

$$2\left(\frac{\pi R^2}{2\pi}\omega - \frac{1}{2}R^2\sin\omega\right) \geq \frac{\pi R^2}{2\pi}\left(\frac{4\pi}{3}\right).$$

Simplification leads to $\omega - \frac{2\pi}{3} \geq sin\omega$. We see that $\omega \geq \frac{2\pi}{3}$, since $\sin\omega \geq 0$. Then,

$$r = R\cos\frac{\omega}{2} \leq R\cos\left(\frac{1}{2}\frac{2\pi}{3}\right) = \frac{R}{2}.$$

Now, consider Fig. 7, where $\mathrm{B}(\mathcal{E})$ intersects $D(S_i; R)$ in all three sectors. Let $\omega \in (0, \pi]$ be the value of $\angle AS_iB$. Then, $\omega \geq 2\pi/3$. Therefore, $r = R\cos\frac{\omega}{2} \leq \frac{R}{2}$. Summarizing the above shows that $r < \frac{R}{2}$.

Similarly, when the random bisection method is used in obtaining $\{\mathcal{NN}\}$, we can also show that $r < \frac{R}{2}$.

Due to the above property of R, we call $R/2$ the *tolerance radius*.

5.3 Evaluation of $\mathcal{C}_3$

Here, we first describe a quantity to judge how well $\mathcal{C}_3$ can be used to fit the boundary. Then, we present a quantity to examine how many sensors that are "far away" are included in $\mathcal{C}_3$. We begin with the following definition:

Definition. *For a positive number r, let $\mathrm{BA}(\mathcal{E}; r)$ denote the set of all points in $\mathcal{R}^2$ such that the distance of each point to the boundary $B(\mathcal{E})$ is at most r. The* degree of fitting *of $\mathcal{C}_3$ is defined as*

$$a(\mathcal{C}_3, r) = \frac{|\mathrm{BA}(\mathcal{E}; r) \cap \mathcal{C}_3|}{|\mathrm{BA}(\mathcal{E}; r) \cap S|}. \quad (5)$$

Intuitively, $\mathrm{BA}(\mathcal{E}; r)$ is a strip with width $2r$, centered around the event boundary. The quantity $a(\mathcal{C}_3, r)$ is expected to provide valuable information on whether or not the detection algorithm performs well in detecting the boundary of the event. The reasoning is explained as follows: Suppose $\mathrm{BA}(\mathcal{E}; r)$ is such that all the sensors in $\mathrm{BA}(\mathcal{E}; r)$ provide a good outline of the boundary $B(\mathcal{E})$. If $a(\mathcal{C}_3, r)$ is large, say, above 90 percent, all of the sensors in $\mathrm{BA}(\mathcal{E}; r)$ that are detected by an event detection algorithm are also expected to provide a good outline of the boundary of the event.

The value of r plays an important role in interpreting $\mathrm{BA}(\mathcal{E}; r)$ and $a(\mathcal{C}_3, r)$. If r is large, say, above $R/2$, Section 5.2 shows that many sensors within $\mathrm{BA}(\mathcal{E}; r)$ will not be detected so that $a(\mathcal{C}_3, r)$ can be very low. On the other hand, if r is very small, $\mathrm{BA}(\mathcal{E}; r)$ may become a strip containing few sensors so that $\mathrm{BA}(\mathcal{E}; r)$ does not present a good description of the boundary $B(\mathcal{E})$. A natural question is then: How can one choose an appropriate r such that $\mathrm{BA}(\mathcal{E}; r)$ provides a good outline of the boundary and $a(\mathcal{C}_3, r)$ is informative?

To get an answer, we first note that if these N sensors are placed into the field by using the standard grid method, a typical grid is a square with width equal to $b/\sqrt{N}$. Given $\mathrm{BA}(\mathcal{E}; r)$, randomly draw a square Q "inside" $\mathrm{BA}(\mathcal{E}; r)$ such that 1) its width is $2r$ and 2) two sides of the square are "perpendicular" to the boundary $B(\mathcal{E})$ (see Fig. 8 for an example of Q).

Set $2r = c \times \frac{b}{\sqrt{N}}$, where c is to be determined. That is, the width of the fitted square Q equals c times the width of a typical grid square. Clearly, the expected number of sensors caught by Q is

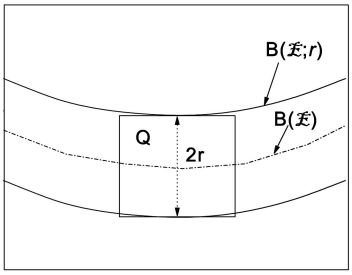


Fig. 8. An illustration of the square Q fitted into the boundary area.

$$N \times \left(\frac{\text{area of } Q}{\text{area of sensor field}}\right) = \frac{N \times \left(c \times \frac{b}{\sqrt{N}}\right)^2}{b^2} = c^2.$$

For $\mathrm{BA}(\mathcal{E}; r)$ to provide a good outline of the boundary, intuitively, we could choose r such that c^2, which is the expected number of sensors inside Q, equals 1. When $c^2 = 1$, r has the following value:

$$r_1 = \frac{1}{2}\left(\frac{b}{\sqrt{N}}\right). \quad (6)$$

Note that r_1 equals half the width of a typical grid.

We now turn to examining how many sensors not close to the boundary are contained in $\mathcal{C}_3$. Motivated by Section 5.2, we only check those sensors whose distances to the boundary are at least $R/2$. Let $\mathrm{A}(\mathcal{E}; R)$ denote the set of all points in $\mathcal{R}^2$ such that the distance of each point to the boundary $B(\mathcal{E})$ is at least $R/2$. Define the *false detection rate* of $\mathcal{C}_3$ to be the following quantity:

$$e(\mathcal{C}_3, R) = \frac{|\mathrm{A}(\mathcal{E}; R) \cap \mathcal{C}_3|}{|\mathrm{A}(\mathcal{E}; R) \cap S|}. \quad (7)$$

If $e(\mathcal{C}_3, R)$ is small, sensors far away from the event boundary are not likely to be contained in $\mathcal{C}_3$.

6 Simulation

In this section, we describe our simulation setup, discuss the issue of determination of threshold values, and report our experimental results.

6.1 Simulation Setup

We use MATLAB to perform all simulations. All the sensor nodes are uniformly distributed in a 64×64 square region. The number of nodes is 4,096. Without loss of generality, we assume that the square region resides in the first quadrant such that the lower-left corner and the origin are co-located. Sensor coordinates are defined accordingly. Normal sensor readings are drawn from $N(\mu_1, \sigma_1^2)$, whereas event sensor readings are drawn from $N(\mu_2, \sigma_2^2)$. In the simulation, we choose $\mu_1 = 10$, $\mu_2 = 30$, and $\sigma_1 = \sigma_2 = 1$. Note that these means and variances can be picked arbitrarily, as long as $|\mu_1 - \mu_2|$ is large enough compared with σ_1 and σ_2. We choose $\sigma_1 = \sigma_2 = 1$ because they represent the system

TABLE 1
Relationship between θ_1 and p When ys Are *Identical Independent Distribution* (i.i.d.) from $N(0,1)$

p	0.26%	5%	10%	15%	20%	25%
θ_1	3.00	1.96	1.65	1.44	1.28	1.15

calibration error, which should be small for a sensor that is not outlying.

In all the simulation scenarios, we choose $\mathcal{N} = \mathcal{N}^*$, and $\mathcal{N}(S_i)$ contains all one-hop neighbors of S_i. Construction of $\mathcal{NN}(S_i)$ is based on $\mathcal{N}(S_i)$. Increasing the size of $\mathcal{N}$ requires increasing either the transmission range, to enlarge the one-hop neighbor set, or the hop count. We note that multihop neighborhood information implies high communication overhead. Since our simulation focuses on the evaluation of the proposed algorithms, we choose to increase the transmission range and, thus, $\mathcal{N}$ always contains one-hop neighbors. We call the average number of sensors in $\mathcal{N}$ the *density* of the sensor network.

To simulate Algorithm 1 for outlying sensor detection, no event is generated in the network region. All outlying values are drawn from $N(30,1)$. In the simulation of event boundary detection, a sensor in the event region gets a value from $N(10,1)$ with probability p and a value from $N(30,1)$ with probability $1-p$. A sensor out of the event region gets a value from $N(30,1)$ with probability p and a value from $N(10,1)$ with probability $1-p$. These settings are selected to make readings from an event region and readings outside the region largely interfere with each other. Though various event regions with different boundary shapes can be considered, in this paper, we focus on two typical cases: the event regions with ellipses or with straight lines as the boundaries. Straight lines are selected because, when the event region is large, the view of a sensor near the boundary is approximated by a line segment in most cases. An ellipse represents a curly boundary. Our simulation produces similar results for event regions with other boundary shapes. The event regions are generated as follows: For a linear boundary, a line $y = kx + b$ is computed, where $k = tan\gamma$ is the slope, with γ drawn randomly from $(0, \frac{\pi}{2})$, and b is the intercept, drawn randomly from $(-16, 16)$. The area below the line is the event region. For a curly boundary, the event region is bounded by an ellipse that can be represented by $E(a,b,x_0,y_0,\nu) = 0$ [6]. Here, $2a$ and $2b$ are the lengths of the major and minor axes of the ellipse, with a and b drawn randomly from [4] and [16]. (x_0, y_0) is the center of the ellipse, where x_0 and y_0 are randomly chosen from $[a, 64-a]$. ν is the angle between the major axis of the ellipse and the x-axis is a random number between 0 and π.

Note that both Algorithms 1 and 2 need thresholds θ_1 and θ_2 to compute $\mathcal{C}_1$ and $\mathcal{C}_2$. Determination of these values is discussed in the next subsection.

6.2 Determination of Thresholds

One possibility of obtaining the threshold values [9] is to estimate them according to p, which is the probability that a sensor becomes outlying. For example, from the theory of normal distributions, the ideal relationship between θ_1 and p, under the strict assumptions stated after Algorithm 1, may be seen partially in Table 1. However, due to practical violations of the assumptions, the actual relationship might deviate significantly from the one listed in the table. Therefore, to obtain more accurate detection results, one needs some alternative methods to estimate the threshold settings. For convenience, we simply choose $\theta_2 = \theta_1 = \theta$ in this paper because both Algorithms 1 and 2 follow the same procedure except that they utilize different neighborhoods to compute d value. This means that once we have an estimate for θ_1, the same estimate will also be used for θ_2. In this section, we propose to determine θ_1 by using an adaptive threshold determination scheme through the ROC (receiver operating characteristic) curve analysis [10].

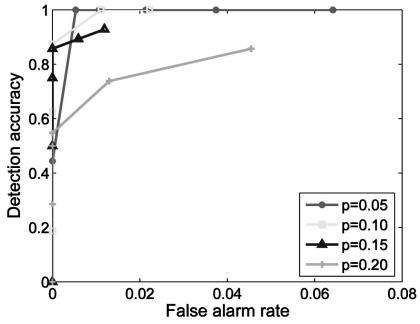


Fig. 9. Examples of ROC curves for $\text{density} = 30$ and different values of p.

Here is the idea of this scheme. Periodically, the base station queries a subregion, where no event occurs, to obtain a training data set and then executes the algorithm for outlying sensor detection based on different threshold settings. We note that the utilization of such training data is practical in the sensor networks [2], [8]. Each time the false-alarm rates and detection accuracies based on one query are used to construct an ROC curve, where the abscissa represents the false alarm rate, and the ordinate denotes the detection accuracy. This curve outlines the relationship between the accuracy and the false alarm rate when varying the threshold values of θ. The well-known fact is that an improvement in the detection accuracy is usually accompanied by an increase in the false alarm rate. A common practical choice of θ is obtained from the knee point on the curve, where the detection accuracy increases limitedly, whereas the false alarm rate exhibits a large boost. The base station will then broadcast the settings of the thresholds to the whole network.

In our simulation studies, we only consider six threshold settings: 3, 1.96, 1.65, 1.44, 1.28, and 1.15, as shown in Table 1. For each query, the training data set contains 196 nodes, that is, about 5 percent of the entire network. In the light of the ratio of the increase in detection accuracy to the increase in the false alarm rate, one of these six settings will be selected as the optimal setting, which corresponds to the knee point. Details are provided as follows.

Fig. 9 shows some ROC curves under network density of 30 and different values of p, where the six points on each

TABLE 2
False Alarm Rates and Detection Accuracies Associated with the Points Indicated on the ROC Curves in Fig. 9

θ	$p = 0.05$	$p = 0.10$	$p = 0.15$	$p = 0.20$
3	0.0000	0.0000	0.0000	0.0000
	0.4444	0.1875	0.0000	0.0000
1.96	0.0053	0.0000	0.0000	0.0000
	1.0000	0.6250	0.5000	0.2857
1.65	0.0107	0.0000	0.0000	0.0000
	1.0000	0.8750	0.7500	0.5000
1.44	0.0214	0.0000	0.0000	0.0000
	1.0000	0.8750	0.8571	0.5476
1.28	0.0374	0.0111	0.0060	0.0130
	1.0000	1.0000	0.8929	0.7381
1.15	0.0642	0.0222	0.0119	0.0455
	1.0000	1.0000	0.9286	0.8571

curve correspond to $\theta = 3, 1.96, 1.65, 1.44, 1.28, 1.15$ in a clockwise order. The detection accuracies and the false-alarm rates associated with the points on the curves are listed in Table 2, where, for a given cell, the first number represents the false alarm rate and the second number the detection accuracy. In the figure, we notice that the detection accuracy increases, accompanied with a higher false alarm rate, along with a smaller threshold setting. This is the trade-off, summarized by ROC curves, between the detection accuracy and the false alarm rate. Determination of the optimal threshold value can be illustrated as follows: Consider $p = 0.15$ in Fig. 9. It is seen that the detection accuracy has an apparent improvement, accompanied with a slight increase in the false alarm rate when we change the threshold from 1.65 to 1.44. (Actually, at this stage, the increase in the false alarm rate is so small that it cannot be seen in either Fig. 9 or Table 2.) Nevertheless, the detection accuracy converts to increase slowly, whereas the false alarm rate exhibits a large boost with the threshold shifting from 1.44 to 1.28. Therefore, the point on the curve that corresponds to the threshold value of 1.44 should be treated as the knee point so that the optimal threshold equals 1.44 when $p = 0.15$. In general, identification of the optimal threshold value can be made by the following vigorous procedure.

First, note that for any given density and value of p, the corresponding ROC curve consists of five line segments. For each line segment, determined by threshold values $\theta^{(1)}$ and $\theta^{(2)}$ $(\theta^{(1)} > \theta^{(2)})$, we can compute the angle (not more than $\pi/2$) formed by this line segment and a horizontal line segment. Examples of these angles are given in Table 3. In the table, the five components under a value of p are angles corresponding to $\theta_1 = 3.00, 1.96, 1.65, 1.44,$ and 1.28, respectively. Clearly, the angle derived from a line segment

TABLE 3
Angles of Line Segments of ROC Curves in Fig. 9

$p = 0.05$	$p = 0.10$	$p = 0.15$	$p = 0.20$
1.5612	1.5708	1.5708	1.5708
0	1.5708	1.5708	1.5708
0	1.5708	1.5708	1.5708
0	1.4821	1.4056	1.5027
0	0	1.4056	1.3045

TABLE 4
Optimal Threshold Values Derived from Table 3

	$p = 0.05$	$p = 0.10$	$p = 0.15$	$p = 0.20$
θ	1.96	1.44	1.44	1.28

depends on the ratio of the increase in detection accuracy to the increase in false alarm rate. Table 3 shows that as $\theta^{(1)}$ decreases, the angle decreases. As the angle becomes smaller, the line segment becomes less steep. Our experience shows that the following is an efficient way to choose the optimal threshold value: If there exists at least one pair $(\theta^{(1)}, \theta^{(2)})$ such that the angle from the corresponding line segment is less than 1.5 (that is, 86 degrees), the largest value of such $\theta^{(1)}$ is selected as the optimal setting of the threshold; otherwise, 1.15 (the smallest among the six preselected values of the threshold setting) is chosen as the optimal setting. Reconsider the case with $\text{density} = 30$ and $p = 0.15$. It follows from Table 3 that only angles from the line segments corresponding to $(\theta^{(1)}, \theta^{(2)}) = (1.44, 1.28)$ and $(1.28, 1.15)$ are less than 1.5. Then, the largest $\theta^{(1)}$ value such that the angle is less than 1.5 is 1.44. Therefore, 1.44 is the optimal threshold value. Optimal threshold values, derived from Table 3, are provided in Table 4. We see that the optimal threshold setting is smaller when there are more outlying sensors in the network, that is, p is larger.

6.3 Simulation Results

In this section, we report our simulation results, each representing an averaged summary of more than 100 runs. More specifically, for each run of computation, the sensor field with or without an event region is generated using the procedure in Section 6.1, the optimal threshold value is determined using the method described in Section 6.2, and then Algorithm 1 or Algorithm 2 is applied to obtain the values of the performance metrics. The averaged values of the performance metrics that are more than 100 runs are reported as our simulation findings. The performance metrics include the detection accuracy and the false alarm rate for outlying sensor detection, as defined by (4) in Section 5.1 for the evaluation of $\mathcal{C}_1$, and the degree of fitting and the false detection rate for event boundary detection, as defined by (5) and (7) in Section 5.3 for the evaluation of $\mathcal{C}_3$. Recall that two sets, $\mathcal{C}_1$ and $\mathcal{C}_3$, contain the detected outlying sensors and boundary sensors, respectively.

We note that, based on the detection accuracy and the false alarm rate, one usually has a good idea on whether or not our outlying sensor detection algorithm works well in practice. However, the use of information on the degree of fitting and false detection rate is somewhat subjective. Sometimes, our algorithm yields a clear outline of the event boundary, but the degree of fitting can be low, and the false detection rate can be high. In this paper, we do not make any effort in trying to determine what values of the degree of fitting and the false detection rate could indicate that the event boundary has been successfully located. The related issue will be addressed in our future work.

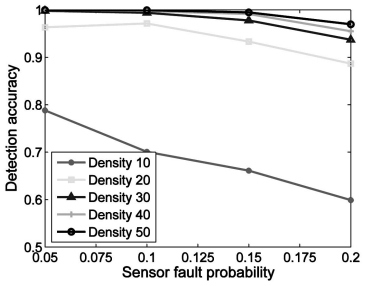


Fig. 10. Detection accuracy versus sensor outlying probability p for different density values.

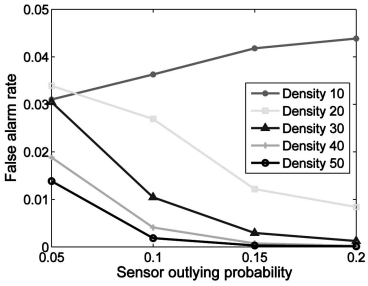


Fig. 11. False alarm rate versus sensor outlying probability p for different density values.

Figs. 10 and 11 plot the detection accuracy and false alarm rate versus p, under different network densities for the outlying sensor detection. Fig. 10 shows a general trend that the detection accuracy decreases as p increases. In Fig. 11, we observe that for densities equal to 20, 30, 40, and 50, the false alarm rate decreases as p increases. By carefully tracing back the intermediate results, we find that the adaptive threshold settings help decrease the false alarm rate. In Figs. 10 and 11, we also observe that a higher network density often leads to higher detection accuracy and a lower false alarm rate. This is reasonable because more sensors in $\mathcal{N}$ and $\mathcal{N}^*$ together bring more information for better results. Note that when $p \leq 0.2$ and density $\geq$ 30, the detection accuracy is above 93.7 percent, and false alarm rate is less than 3.1 percent. For $\text{density} = 20$, the detection accuracy is around 90 percent, and false alarm rate is less than 3.5 percent. In both graphs, we observe that a smaller number of sensors in $\mathcal{N}$ with $\text{density} = 10$ may not be a good choice for Algorithm 1.

For the event boundary detection described in Section 4, we need to set m in transition from $\mathcal{C}_1$ and $\mathcal{C}_2$ to $\mathcal{C}_3$. Through simulation studies, we observe that a larger m usually results in a higher degree of fitting and a higher false detection rate when the network density and p are fixed. For convenience, we fix $m = 4$ in the following, since this setting, in general, achieves a good degree of fitting and a low false detection rate. Plots of the experimental results for various scenarios are shown in Fig. 12.

The main observations from these plots are summarized as follows: 1) For a fixed density, as p increases, the degree of fitting tends to decrease, and the false detection rate tends to increase. The main reason for this is that for a larger p, more outlying sensors interfere with boundary nodes. 2) For a fixed sensor outlying probability, as density increases, the degree of fitting tends to become larger, and the false detection rate tends to become smaller. This is due to the fact that, with a higher density, more information is available for the detection algorithms. 3) For a given shape (line or ellipse) of the event boundary, the random trisection method outperforms the random bisection method. This is because random trisection induces a larger $\mathcal{NN}$ so that more information is utilized in the detection process.

In Fig. 12, we see that the degree of fitting is low for cases where $\text{density} = 10$. Note that a low degree of fitting does not mean that the boundary cannot be detected. Instead, it means that more sensors close to the boundary escape the detection. For example, Fig. 5d indicates a case where the degree of fitting is as low as 53 percent for $p = 0.2$. However, for such a low value of the degree of fitting, the elliptical boundary is still clearly identified. (In this scenario, the outlying sensor detection accuracy is 92 percent.)

We have also conducted simulations when input data are binary decision predicates and obtained results close to those reported in Fig. 12. This indicates that our algorithms are applicable to both 0/1 decision predicates and numeric sensor readings.

7 Conclusions and Future Work

In this paper, we present localized algorithms for the identification of outlying sensors and event boundary in sensor networks. We also propose a scheme to adaptively adjust the thresholds in the base station. Simulation results indicate that these algorithms can clearly detect the event boundary and can identify outlying sensors with a high accuracy and a low false alarm rate when as many as 20 percent of the outlying readings exist.

We believe that our ideas in detecting outlying sensors and event boundaries can be extended to multimodality sensor networks, and the data aggregation can be done along both temporal and spatial dimensions for decreasing the false alarm rate in outlying sensor detection and the false detection rate for boundary detection. Thus, we propose to explore along these directions in the future. We will further study the minimum density requirement for an expected detection accuracy of our proposed algorithms, which can be used as a guideline for the selection of the neighborhood $\mathcal{N}$ and/or $\mathcal{N}^*$ in real network scenarios.

Furthermore, we intend to study other techniques for thresholds (both θ_1 and θ_2) computation. For example, θ values may be determined based on the minimization of a cost-based system. We will conduct more simulations for the case of $\theta_1 \neq \theta_2$ to study the performance of the proposed algorithms. We also plan to study the robust estimates for the population standard deviation. These estimates will replace the corresponding estimates, such as $\hat{\sigma}$, in (2) in the present description of our algorithms. Robust estimates are less influenced by the values of outlying sensors. A potential benefit of the use of robust estimates of population standard deviation is that we could use the percentiles of the standard normal distribution or χ_1^2 to make decisions, instead of the thresholds.

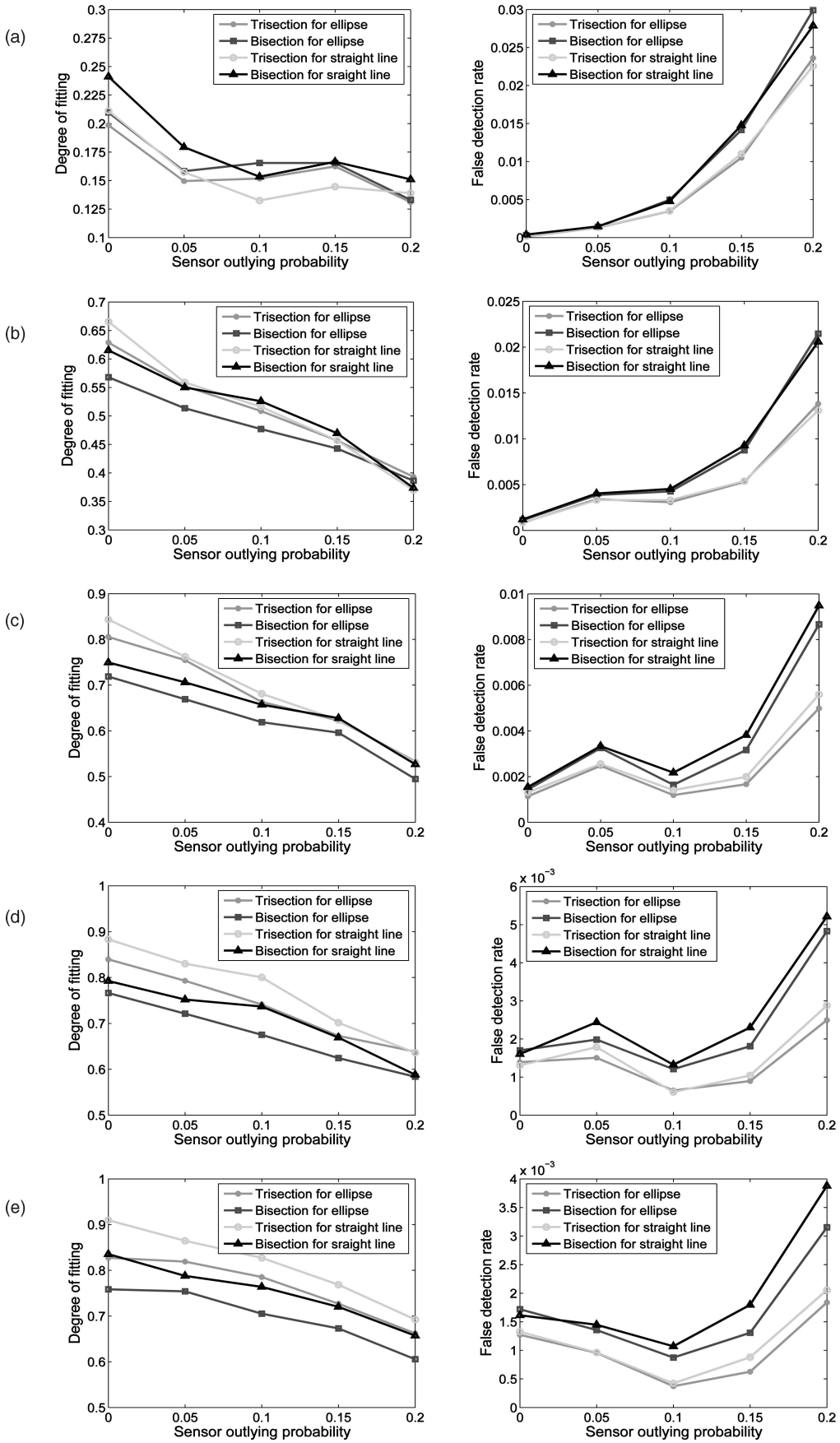


Fig. 12. Performance of the event boundary detection algorithm under different values of sensor outlying probability p and network density. The first, second, third, fourth, and fifth rows have a density equal to 10, 20, 30, 40, and 50, respectively.

ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation under NSF Grant ACI-0305567 and in part by NSF CAREER Award CNS-0347674.

REFERENCES

[1] V. Barnet and T. Lewis, *Outliers in Statistical Data.* John Wiley and Sons, 1994.
[2] R. Brooks, P. Ramanathan, and A. Sayeed, "Distributed Target Classification and Tracking in Sensor Networks," *Proc. IEEE,* vol. 91, no. 8, pp. 1163-1171, Aug. 2003.
[3] D. Chen and X. Cheng, "An Asymptotic Analysis of Some Expert Fusion Methods," *Pattern Recognition Letters,* vol. 22, pp. 901-904, 2001.
[4] D. Chen, X. Cheng, and M. Ding, "Localized Event Detection in Sensor Networks," manuscript, 2004.
[5] X. Cheng, A. Thaeler, G. Xue, and D. Chen, "TPS: A Time-Based Positioning Scheme for Outdoor Sensor Networks," *Proc. IEEE INFOCOM '04,* Mar. 2004.
[6] K.K. Chintalapudi and R. Govindan, "Localized Edge Detection in Sensor Fields," *IEEE Ad Hoc Networks J.,* pp. 59-70, 2003.
[7] T. Clouqueur, K.K. Saluja, and P. Ramanathan, "Fault Tolerance in Collaborative Sensor Networks for Target Detection," *IEEE Trans. Computers,* vol. 53, no. 3, pp. 320-333, Mar. 2004.
[8] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong, "Model Driven Data Acquisition in Sensor Networks," *Proc. 30th Int'l Conf. Very Large Data Bases (VLDB '04),* Aug. 2004.
[9] M. Ding, D. Chen, K. Xing, and X. Cheng, "Localized Fault-Tolerant Event Boundary Detection in Sensor Networks," *Proc. IEEE INFOCOM '05,* Mar. 2005.
[10] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification.* Wiley-Interscience, 2000.
[11] B. Krishnamachari and S. Iyengar, "Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks," *IEEE Trans. Computers,* vol. 53, no. 3, pp. 241-250, Mar. 2004.
[12] K. Krivoruchko, C. Gotway, and A. Zhigimont, "Statistical Tools for Regional Data Analysis Using GIS," *Proc. 11th ACM Int'l Symp. Advances in Geographic Information Systems (GIS '03),* pp. 41-48, Nov. 2003.
[13] D. Li, K.D. Wong, Y.H. Hu, and A.M. Sayeed, "Detection, Classification, and Tracking of Targets," *IEEE Signal Processing Magazine,* vol. 19, pp. 17-29, Mar. 2002.
[14] F. Liu, X. Cheng, D. Hua, and D. Chen, "Range-Difference-Based Location Discovery for Sensor Networks with Short-Range Beacons," to be published in *Int'l J. Ad Hoc and Ubiquitous Computing,* 2007.
[15] C.T. Lu, D. Chen, and Y. Kou, "Detecting Spatial Outliers with Multiple Attributes," *Proc. 15th Int'l Conf. Tools with Artificial Intelligence,* pp. 122-128, Nov. 2003.
[16] C.T. Lu, D. Chen, and Y. Kou, "Algorithms for Spatial Outlier Detection," *Proc. Third IEEE Int'l Conf. Data Mining,* pp. 597-600, Nov. 2003.
[17] T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos, "Distributed Deviation Detection in Sensor Networks," *Sigmod Record,* vol. 32, no. 4, pp. 77-82, Dec. 2003.
[18] J. Staddon, D. Balfanz, and G. Durfee, "Efficient Tracing of Failed Nodes in Sensor Networks," *Proc. First ACM Workshop Wireless Sensor Networks and Applications (WSNA '02),* pp. 122-130, Sept. 2002.
[19] S. Tanachaiwiwat, P. Dave, R. Bhindwale, and A. Helmy, "Location-Centric Isolation of Misbehavior and Trust Routing in Energy-Constrained Sensor Networks," *Proc. IEEE Workshop Energy-Efficient Wireless Comm. and Networks (EWCN '04),* Apr. 2004.
[20] A. Thaeler, M. Ding, X. Cheng, and D. Chen, "iTPS: An Improved Location Discovery Scheme for Sensor Networks with Long Range Beacons," *J. Parallel and Distributed Computing,* vol. 65, no. 2, pp. 98-106, Feb. 2005.
[21] S.S. Wilks, *Mathematical Statistics.* John Wiley and Sons, 1962.

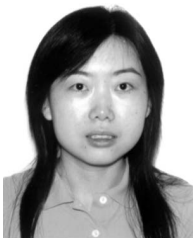
Weili Wu received the MS and PhD degrees in computer science from the University of Minnesota in 1998 and 2002, respectively. She is currently an assistant professor and the laboratory director of the Database Research Laboratory, Department of Computer Science and Engineering, University of Texas, Dallas. She is an editor of the research monograph *Clustering and Information Retrieval*. Her research interest is mainly in database systems, especially in spatial database with applications in geographic information systems and bioinformatics, distributed database in Internet system, and wireless database systems in connection to wireless communication. She has published more than 30 research papers in various prestigious journals and conferences such as the *IEEE Transactions on Multimedia*, *Theoretical Computer Science*, *Journal of Complexity*, *Discrete Mathematics*, *Discrete Applied Mathematics*, the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, the SIAM Conference on Data Mining, the University Consortium of Geographic Information Sciences (UCGIS) Summer Assembly, and the International Conference on Computer Science and Informatics. She is an author of the textbook *Mathematical Theory of Optimization*. She is a member of the IEEE, the IEEE Computer Society, and the ACM.

Xiuzhen Cheng received the MS and PhD degrees in computer science from the University of Minnesota, Twin Cities, in 2000 and 2002, respectively. She is an assistant professor in the Department of Computer Science, George Washington University. She has served in the editorial boards of technical journals including the *Ad Hoc Networks Journal*, the *International Journal on Ad Hoc and Ubiquitous Computing*, and the *International Journal of Sensor Networks*. She was the program cochair of the First International Conference on Wireless Algorithms, Systems, and Applications (WASA '06). She has served as a Transaction Performance Processing Council (TPC) member for various professional conferences such as the IEEE INFOCOM, the IEEE International Conference on Mobile Ad Hoc and Sensory Systems (MASS), and so forth. She worked as a program director at the US National Science Foundation for six months in 2006. Her current research interests include wireless and mobile computing, sensor networks, wireless security, statistical pattern recognition, approximation algorithm design and analysis, and computational medicine. She received an US National Science Foundation Faculty Early Career Development (CAREER) award in 2004. She is a member of the IEEE.

Min Ding received the BS degree in engineering from Tianjin University in 1996 and the MS degree in information science from the Institute of Scientific and Technical Information of China in 1999. From August to December 1999, she was a visiting student in the College of Information Studies, University of Maryland, College Park. Currently, she is a PhD student in the Department of Computer Science, George Washington University. Her research interests include various topics in localized information processing in sensor networks, with a focus on fault-tolerant algorithm design. She is a student member of the IEEE.

Kai Xing received the BSc degree in computer science from the University of Science and Technology of China in 2003. Currently, he is a PhD candidate at the Department of Computer Science, George Washington University. His current research focuses on mobile and wireless communication and networks. He is a student member of the IEEE.

Fang Liu received the BS degree in computer science from the University of Science and Technology of China in 2000 and the MS degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, in 2003. She is currently a PhD candidate in the Department of Computer Science, George Washington University. Her current research interests include wireless security, localization and tracking, and information retrieval in sensor networks. She is a student member of the IEEE.



Ping Deng received the master's degree in computer science from the University of Texas, Dallas, in August 2003. She is a PhD candidate in computer science at the same university. Her research interests include database systems, data mining, and bioinformatics. She is a student member of the IEEE.