Localized Fault-Tolerant Event Boundary Detection in Sensor Networks

Min Ding
Computer Science
The George Washington University
Washington, DC 20052, USA
minding@gwu.edu

Dechang Chen
Uniformed Services University
of the Health Sciences
Bethesda, MD 20817, USA
dchen@usuhs.mil

Kai Xing & Xiuzhen Cheng
Computer Science
The George Washington University
Washington, DC 20052, USA
{kaix,cheng}@gwu.edu

Abstract— **This paper targets the identification of faulty sensors and detection of the reach of events in sensor networks with faulty sensors. Typical applications include the detection of the transportation front line of a contamination and the diagnosis of network health. We propose and analyze two novel algorithms for faulty sensor identification and fault-tolerant 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 faulty sensors with a high accuracy and a low false alarm rate when as many as 20% sensors become faulty.**

Our work is exploratory in that the proposed algorithms can accept any kind of scalar values as inputs, a dramatic improvement over existing works that take only 0/1 decision predicates. Therefore, our algorithms are generic. They can be applied as long as the "events" can be modelled 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.

Keywords: Sensor networks, event boundary detection, fault tolerance, faulty sensor identification.

I. Introduction

The marriage of sensor and network technologies provides many exciting applications of sensor networks [4], [17], [27]. With their capabilities for monitoring and control, networked sensors are expected to be widely deployed. Such a network can provide a fine global picture through the collaboration of many sensors with each observing a coarse local view [13], [16]. The difficulty in flourishing these networks lies in in-network processing observations from sensors in close geographic proximity [8].

One important task of a typical sensor network is to monitor, detect, and report the occurrences of interesting events (e.g. forest fire, chemical spills, etc.) with the presence of *faulty* sensor measurements. These events usually span some geographic region and in many application scenarios the detection of the event boundary may become more important than the detection of the entire event region. A good example is the timely estimation of the possible reach of the contamination in a surveillance network monitoring the transportation of chemical spills in soil. On the other hand, individual sensor reading is not reliable. Filtering out faulty readings and transmitting only the boundary information to the base station can save energy. In this paper we target the problem of identifying faulty sensors and detecting event boundaries in sensor networks with faulty sensors.

There exist other application scenarios that motivate our research. In sensor network health diagnosis [28], we would like to identify the regions containing sensors that behave differently from those in the outside of the regions. For example, in military sensor networks, enemies can destroy or attack the sensors within a reachable region. In habitat monitoring, sensors in close proximity may have a similar level of residual energy since they are on the same duty [28]. In both cases, most or all sensors within some region may be dead, or behave arbitrarily or maliciously. A typical event region detection may fail due to the lack of meaningful sensor readings, and computing the boundary then becomes a feasible alternative.

However, faulty sensor detection and event boundary computation are not trivial at all. One significant challenge facing the two tasks comes from the strict resource limitation (battery power, bandwidth, etc.). Sensors are powered by battery, which may not be recharged or replaced after deployment. We can not afford to allow a base station collect all sensor measurements and compute event boundaries in a centralized fashion [9], [10], [16]. Therefore, we have to seek localized and computationally efficient algorithms for each node to determine whether it is faulty or whether it is close to the event boundary. The presence of faulty sensors constitutes

another significant challenge for event boundary computation. Sensor readings may be faulty due to hardware crash, security attack, or environment disturbance. Thus a solid event boundary detection algorithm must be robust and fault-tolerant.

Our major contribution consists of one localized algorithm for faulty sensor identification and one localized algorithm for fault-tolerant event boundary detection. These two algorithms are generic. They can take as inputs any type of numbers provided by sensors. This is fundamentally different from the existing works in event, event region, and event boundary computation [6], [7], [11], where sensors are required to access a binary decision procedure by comparing their readings with a predetermined threshold. This binary decision is called a *0/1 decision predicate*, with 1 indicating an event status and 0 indicating a normal status. Using 0/1 decision predicates for event, event region, and event boundary computation may have the following disadvantages:

- 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, 0/1 predicates will miss the spatial information on deployed sensors.
- 0/1 decision predicates are the preprocessing results of the actual measured data. Detection over binary predicates represents the second round approximation.
- 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.

There are some other unique characteristics of the proposed algorithms. The boundary detection algorithm takes into account the issue of *fault-event disambiguation.* We note that both faulty sensors and sensors in an event region can generate "abnormal readings" deviated from a typical application-specific range. Intuitively, when a remarkable change in sensor readings is detected, something must have happened. If the change is present with a single sensor only, the sensor is faulty. If most neighboring sensors observe the same phenomenon simultaneously, an event occurs. In other words, readings from faulty sensors are geographically independent. But readings from sensors in close proximity are spatially correlated [11]. This property has been embedded into our event boundary detection algorithm. Simulation results indicate that our algorithms have a strong fault-tolerant ability and are effective in identifying faulty sensors and locating the event boundaries. For example, for a moderately dense sensor network (about 30 nodes in the neighborhood) with 20% faulty sensors, our algorithms can clearly detect the event boundary and maintain a high faulty sensor detection accuracy (above 90%) and a low false alarm rate.

Our approach to faulty sensor and event boundary detection is related to the outlier detection and regional data analysis in spatial data mining [12], [14], [15]. It is expected that the algorithms proposed in this paper will find many applications in fields other than sensor networks.

This paper is organized as follows. We first briefly summarize the related work in Section II and explain useful notations in Section III. The two localized algorithms for faulty sensor identification and fault-tolerant event boundary detection are proposed in Sections IV and V. Performance metrics and analysis are outlined in Section VI. Our simulation results are reported in Section VII. We conclude our paper in Section VIII with future work discussion.

II. Related Work

In this section, we summarize related works along three major lines: 0/1 decision predicate computation, faulty sensor detection, and event region and event boundary detection. Most existing deviation and event detection algorithms rely on the 0/1 decision predicates computed by individual sensors.

As we have noted earlier, when a remarkable change in the readings of sensors is detected, a faulty sensor or a real event must have occurred. This observation is explored in [3], [13], [19] 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 are exploited. In [3], the "change point" of the time series are statistically computed. The result can be used to answer questions such as "when does the front line of the contamination reach location (x, y)?" The detector proposed in [13] computes a running average and compares it with a threshold, which can be adjusted by false alarm rate. In [19], *kernel density estimators* are designed to check whether the number of "abnormal" readings are beyond an application-specific threshold. Note that none of these works can disambiguate faulty sensors and real event sensors since only observations from individual sensors are studied. By exploring the correlation among neighboring sensors, our work can tell faulty sensors from event sensors and compute the boundary of the event region.

Failed or misbehaving nodes are faulty sensors that can be detected through route discovery and update. Common routing protocols leverage the re-establishment

of route discovery to evade failed nodes [20]. Staddon, Balfanz, and Durfee [22] propose to trace failed nodes in sensor networks at a base station, assuming 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 [18], nodes can listen-in on the neighbor to detect failed or misbehaving neighbors. In [23], 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 can detect many kinds of misbehaving nodes, as long as the "abnormal behavior" can be modelled by real numbers. Our algorithm does not rely on any routing or global topology information, thus provides better scalability and flexibility. Moreover, our algorithm can be combined with any routing protocol to route the detected information to base stations for further instructions.

Clouqueur, Saluja, and Ramanathan [7] seek algorithms to collaboratively detect the presence of a target in a region G. Each sensor obtains the target energy (or local decision) from all other sensors in the region, drops extreme values if faulty sensors exist, computes the average, and then compares it with a pre-determined threshold for final decision. Correspondingly, the algorithm is termed *value fusion* if the input is target energy, or *decision fusion* if the input is the local decision made by each sensor. Under these algorithms, all sensors in region G will conclude with a same decision. However, both target energy and local decision need to be computed ahead of time. Further, these algorithms do not specify how to define region G, as the communication and computation overheads are strongly related to the size of G. Our algorithm requires only raw readings from the neighborhood and captures the boundary sensors surrounding the region.

Krishnamachari and Iyengar [11] propose several localized threshold based decision schemes to detect both faulty sensors and event regions. The 0/1 decision predicates from the neighborhood are collected and the number of neighbors with the same predicates are calculated. This number is used for the final decision based on a majority vote. The algorithm presented in this paper identifies both faulty sensors and front lines of event regions. It works well with not only 0/1 decision predicates but also numbers that abstract sensor readings or sensor behaviors.

The unique work that targets localized boundary detection in sensor networks is proposed in [6]. All of the three schemes in [6] take as inputs the 0/1 decision predicates from neighboring sensors. The statistical approach computes the number of 0's and 1's in the neighborhood and a boundary sensor is detected if its neighbors contain a "similar" number of 0's and 1's. Here the "similarity" is defined based on a threshold whose value can be obtained based on a lookup table. The image processing approach computes a weighted average of all the neighboring values. The classifier-based approach computes an optimal classifier by sampling the neighboring area. The first two approaches need a threshold whose value is determined by multiple parameters. The classifier-based approach does not need a threshold, but the fine sample granularity contributes a high computational overhead. Our boundary detection algorithm does not need the pre-computation of the 0/1 decision predicates. It is computationally simpler than the last two approaches. Further, the threshold in our algorithm only depends on the fault tolerance requirement.

Network health scans are also related to boundary detection. Residual energy scan (eScan) [28], for example, is proposed to identify regions with low energy supply. This can help users decide when to deploy new sensors to avoid network malfunctions caused by energy depletion. In eScan, a data disseminating tree rooted at the base station is constructed and residual energies from sensors along the tree are aggregated based on some predetermined rules. Our algorithm can obtain similar scans, as long as residual energies have the regional property. However, it is more robust, as no tree or other fragile structures are needed in the algorithm execution.

III. Notations and Network Model

Throughout this paper, we assume that N sensors are uniformly deployed in a $b \times b$ squared field located in the two dimensional Euclidean plane $\mathcal{R}^2$. A sensor's reading is *faulty* (abnormal) if it deviates significantly from other readings of neighboring sensors [1]. Sensors with faulty readings are called faulty sensors. In this paper, the ith sensor S_i and its location will be used exchangeably. We use S to denote the set of all the sensors in the field and R denote 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. For example, a rogue node that continues to inject messages to the network or drop all relay messages in DOS attack is a misbehaving node. 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 readings of the sensors in $\mathcal{E}$ are significantly different from those of sensors not in $\mathcal{E}$. A faulty sensor can be viewed as a special event

which contains only one point, i.e., the sensor itself. A point $\mathbf{x} \in \mathcal{R}^2$ is said to be *in the boundary of* of $\mathcal{E}$ if and only if each closed disk centered at $\mathbf{x}$ contains both points in $\mathcal{E}$ and points not in $\mathcal{E}$. The *boundary* of the event $\mathcal{E}$, denoted by $\mathrm{B}(\mathcal{E})$, is the collection of all the points in the boundary of $\mathcal{E}$. As an example, a circle is the boundary of the region bounded by the circle if the region is an event.

We assume each sensor can compute its physical position through either GPS or some GPS-less techniques such as [5], [21], [24], [26]. Note that in this paper we focus on the fault tolerant event boundary detection, thus the delivery of the event boundaries will not be considered.

IV. Localized Faulty Sensor Detection

In this section, we describe our algorithm for detecting sensors whose readings (measurements) are faulty.

A. Derivation of Detection Procedure

The procedure of locating a faulty sensor could be formalized statistically as follows. Consider how to compare the reading at S_i with those of its neighbors. Let $\mathcal{N}(S_i)$ denote a bounded closed set of $\mathcal{R}^2$ that 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 med_i in equation (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 can not represent well the "center" of a sample 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 faulty. Now we start to quantify the degree of extremeness of d_i. To do this, the differences d from sensors near S_i are needed.

Consider another bounded closed set $\mathcal{N}^*(S_i) \subset \mathcal{R}^2$ that 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 Figure 1 for an illustration of $\mathcal{N}$ and $\mathcal{N}^*$. According to equation (1) sensors in $\mathcal{N}^*(S_i)$ yield d_1, $\cdots$, d_i, $\cdots$, d_n. Now if d_i is extreme in $D = \{d_1, \cdots, d_i, \cdots, d_n\}$, S_i will be treated as a faulty sensor. The decision can be made vigorously using the following procedure. Let $\hat{\mu}$ and $\hat{\sigma}$ denote, respectively, the sample mean and sample standard deviation of the set D, i.e.,

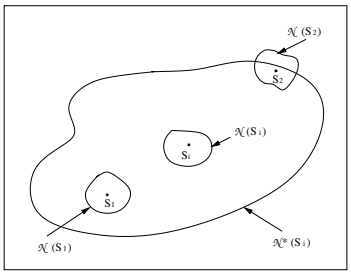


Fig. 1. $\mathcal{N}^*$ neighborhood $\mathcal{N}^*(S_i)$ of sensor S_i and $\mathcal{N}$ neighborhoods of sensors inside $\mathcal{N}^*(S_i)$. Each $\mathcal{N}$ neighborhood is used to compute d_i, while the $\mathcal{N}^*(S_i)$ is used to compare the d_i's.

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

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

Standardize the dataset 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 $|y_i| \geq \theta$, treat S_i as a faulty sensor. Here $\theta(> 1)$ is a preselected number.

We now start to justify 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 $x_1^{(i)}$, $x_2^{(i)}$, $\cdots$, $x_k^{(i)}$ 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 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 equation (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, we first note the following special case of Theorem 9.6.5 in [25]: if kp_k is a positive integer such that $p_k = 0.5+O(1/k)$, then as $k \to \infty$, $x^{(i)}_{(kp_k)}$ converges in probability to $\tilde{\mu}$. For any real number a, let $\lfloor a \rfloor$ denote the largest integer 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

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

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

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

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 [2].

Now consider the following simple scenario where i) readings of sensors in $\mathcal{N}^*(S_i)$, i.e., $x_1, \cdots, x_n$, are independent; ii) 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; iii) 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 i)-iii) that as k becomes large, the sequence $d_1, \cdots, d_n$ form approximately a sample from a normal distribution with mean equal to 0. This implies that as k is large, the sequence from standardization, i.e., $y_1, \cdots, y_n$, 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 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 faulty. When $\theta = 2$, the probability of observing a y_i with $|y_i| \geq 2$ is about 5%.

B. 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 faulty by the above procedure. Note that when an event occurs, a sensor $S_i \in \mathcal{C}_1$ may be close to the event boundary but is not faulty (See Figure 2(a)). The procedure in part IV-A can be summarized into the following algorithm.

ALGORITHM 1

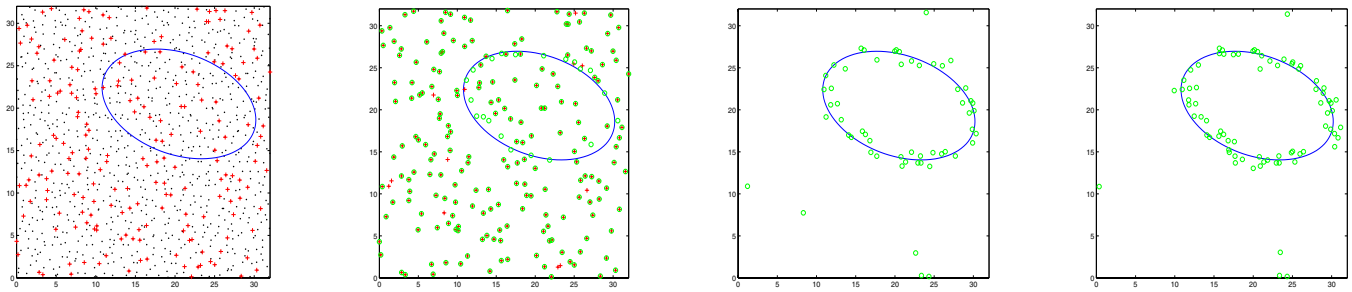
1) Construct $\{\mathcal{N}\}$ and $\{\mathcal{N}^*\}$. For each sensor S_i, perform the following steps.
2) Use $\{\mathcal{N}(S_i)\}$ and equation (1) to compute d_i for sensor S_i.
3) Use $\{\mathcal{N}^*(S_i)\}$ and equation (2) to compute y_i for sensor S_i.
4) If $|y_i| \geq \theta$, where $\theta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

Clearly, $|\mathcal{C}_1|$, the size of $\mathcal{C}_1$, depends on θ. Assuming the y values in equation (2) constitute a sample of a standard normal distribution and the decisions are made independently, then if θ is such 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 faulty if i) data measurement or data collection makes errors, or ii) some variability in the area surrounding the sensor has changed significantly, or iii) the inherent function of the sensor is abnormal. In any of the three cases, readings from faulty sensors do not reflect reality, so that they can be discarded before further analysis on sensor data. However, faulty 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 faulty sensors.

V. 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. As mentioned above, $\mathcal{C}_1$ 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 compared to $|\mu_1 - \mu_2|$. $\{\mathcal{N}\}$ and $\{\mathcal{N}^*\}$ are constructed using closed disks. Consider sensor S_i. 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



(a) Data (b) $\mathcal{C}_1$ (c) $\mathcal{C}_2$ (d) $\mathcal{C}_3$

Fig. 2. 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 Figs 13 and 14. The interior of the ellipse is the event region. A sensor becomes faulty with probability $p = 0.2$. A $\cdot$ represents a sensor and a $+$ represents a faulty sensor. A $\circ$ in (b), (c), and (d) represents a node in $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$, respectively.

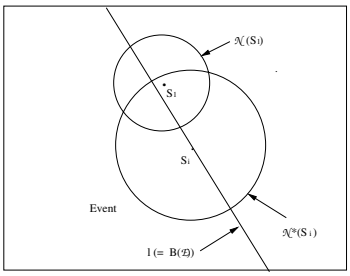


Fig. 3. Event $\mathcal{E}$ is the union of line l and the portion on the left hand side of l. S_i is a sensor located on B($\mathcal{E}$) and S_1 is a sensor inside $\mathcal{N}^*(S_i)$. Both $\mathcal{N}^*(S_i)$ and $\mathcal{N}(S_1)$ are closed disks.

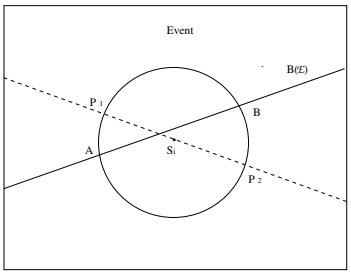


Fig. 4. Illustration of random bisection. $\mathcal{NN}(S_i)$ is the half disk containing P_1, P_2, B, and S_i.

of sensors, calculation based on equation (1) shows that each d follows $N(0, \sigma^2)$ approximately. For example, at sensor S_1 shown in Figure 3, d_1 follows approximately $N(0, \sigma^2)$. The reasoning is 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 0. This shows that S_i can not be detected 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, when S_i close to the boundary can not be detected, 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.

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 two 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, consult Figure 4. 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 the other d values from $\mathcal{N}^*(S_i)$. Then perform calculation in (2) and make a decision on S_i.

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 Figure 5. Form a union using any two sectors and calculate d_i in each union (total =3). The union resulting in the largest $|d_i|$ is $\mathcal{NN}(S_i)$. It is

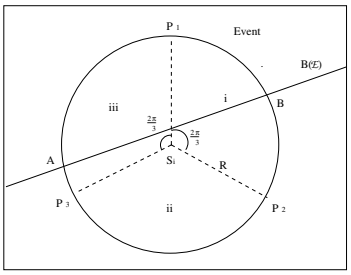


Fig. 5. Illustration of random trisection. Sectors $P_1P_2S_i$, $P_2S_iP_3$, and $P_3S_iP_1$ are numbered as i, ii, and iii, respectively. Each sector contains an angle equal to $2\pi/3$. $\mathcal{NN}(S_i)$ is the union of sectors i and iii.

easy to see that in Figure 5, $\mathcal{NN}(S_i)$ is the union of sectors i and iii. The d_i with the largest $|d_i|$ will replace previous d_i, keeping unchanged all the other d values from $\mathcal{N}^*(S_i)$, and subsequently a decision will be made on S_i.

There are two options of 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. 2(c)). In general, $\mathcal{C}_2$ catches more sensors near the event boundary than $\mathcal{C}_1$ does. Now we discuss how to 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 faulty 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 that are not faulty, 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 to 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 $\mathcal{C}_2$ that 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 $D(S_i; \sqrt{\frac{mb^2}{\pi N}}\ )$ 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 Figure 2(d)). For convenience, sometimes we will write $\mathcal{C}_3$ for $\mathcal{C}_3(m)$.

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 $\mathcal{C}_1$ $(\theta = \theta_1)$.
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 the other d values from $\mathcal{N}^*(S_i)$ obtained in step 1). Use equation (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 the algorithm. 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 the y values in equation (2) constitute a sample of a standard normal distribution and the decisions are made independently, then 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 $\mathcal{C}_3$.

VI. Performance Evaluation

Evaluation of the proposed algorithms include 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 Algorithm 2. The finding is then used to define metrics to evaluate the performance of $\mathcal{C}_3$.

A. 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 number of faulty sensors detected to the total number of faulty sensors, and the *false alarm rate* $e(\mathcal{C}_1)$, defined to be the ratio of number of non-faulty sensors that are claimed as faulty to the total number of non-faulty sensors. Let $\mathcal{O}$ denote the set of faulty sensors in the field, then

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

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

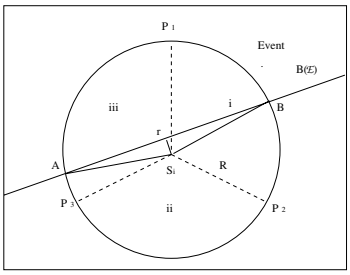


Fig. 6. The event boundary intersects the disk $D(S_i; R)$ in two sectors i and iii. A and B are two intersection points between the event boundary and the boundary of the disk. r is the distance from sensor S_i to the event boundary.

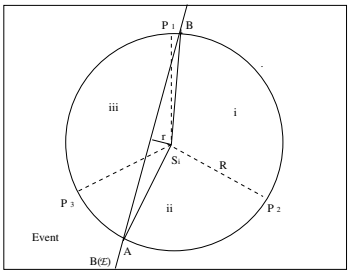


Fig. 7. The event boundary intersects the disk $D(S_i; R)$ in three sectors i, ii, and iii. A and B are two intersection points between the event boundary and the boundary of the disk. r is the distance from sensor S_i to the event boundary.

B. 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 Figure 6 and Figure 7, where $\mathrm{B}(\mathcal{E})$ intersects $D(S_i; R)$ in at least two sectors.

Consider Figure 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, i.e., the part containing P_1, has an 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 $\angle AS_iB$. And consequently, the area of the other part is

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

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

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

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 Figure 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 $\omega \geq 2\pi/3$. So $r = R\cos\frac{\omega}{2} \leq \frac{R}{2}$. 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*.

C. 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 to be

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

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 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%, then all 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.

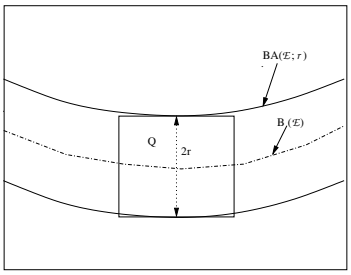


Fig. 8. An illustration of the square Q fitted into the boundary area.

The value of r plays an important role in interpreting $\text{BA}(\mathcal{E};r)$ and $a(\mathcal{C}_3,r)$. If r is large, say, above $R/2$, then the above Section VI-B shows that many sensors within $\text{BA}(\mathcal{E};r)$ will not be detected so that $a(\mathcal{C}_3,r)$ can be very low. On the other hand, if r is very small, $\text{BA}(\mathcal{E};r)$ may become a strip containing few sensors so that $\text{BA}(\mathcal{E};r)$ 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 $\text{BA}(\mathcal{E};r)$ 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 using the standard grid method, then a typical grid is a square with width equal to $b/\sqrt{N}$. Given $\text{BA}(\mathcal{E};r)$, randomly draw a square Q "inside" $\text{BA}(\mathcal{E};r)$ such that i) its width is $2r$; ii) two sides of the square are "perpendicular" to the boundary $B(\mathcal{E})$. See Figure 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

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

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

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

Note that r_1 equals half 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 VI-B, we only check those sensors whose distances to the boundary are at least $R/2$. Let $\text{A}(\mathcal{E};R)$ 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

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

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

VII. Simulation

A. Simulation Set-Up

MATLAB is used to perform all simulations. The sensor network contains 1024 nodes in a square region of size 32×32 units with one sensor randomly placed within each unit grid. Without loss of generality, we assume 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 $N(\mu_1,\sigma_1^2)$ while event sensor readings are drawn from $N(\mu_2,\sigma_2^2)$. In the simulation we choose $\mu_1=10$, $\mu_2=30$, $\sigma_1=\sigma_2=1$. 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 calibration error which should be small for a sensor that is not faulty.

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

Note that both algorithms 1 and 2 need thresholds ($\theta=\theta_1$ to compute $\mathcal{C}_1$ and $\theta=\theta_2$ to compute $\mathcal{C}_2$) to make decisions. In all the simulation scenarios for event boundary detection, we choose $\theta=\theta_1=\theta_2$. The settings of the threshold, listed in Table I, depend on p, the probability that a sensor becomes faulty. If $p=0$, we simply set $\theta=2$ in order to detect the event boundary. The θ values are obtained from a standard normal distribution, as shown in Sections IV and V. Note

p	5%	10%	15%	20%
θ	1.96	1.65	1.44	1.28

TABLE I
THE SETTINGS OF THRESHOLD θ.

that in general p is not *a priori* for a sensor, but its empirical value may be available based on the elapsed time after deployment. On the other hand, sensors can obtain information of the false alarm rate from base stations and adaptively adjust the settings of thresholds.

In our simulation we choose to use the values listed in Table I for simplicity.

To simulate Algorithm 1 for faulty sensor detection, no event is generated in the region. All faulty values are drawn from $N(30, 1)$. In the simulation of event boundary detection, an event region with different boundary shapes, such as a rectangle, a triangle, a circle, an ellipse, a straight line, a sine curve, and so on, are considered. A sensor in the event region gets a value from $N(10, 1)$ with probability p and a sensor out of the event region gets a value from $N(30, 1)$ with probability p. These settings are selected to make readings from an event region and readings outside the region largely interfere with each other. When inputs are binary decision predicates, a sensor flips its value from 1 to 0 or from 0 to 1 with probability p.

We will report the results for event regions with ellipses or straight lines as the boundaries. Straight lines are selected because when the network area is large, the view of the boundary of one 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 $k = tan\ \theta$ is the slope, with θ 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 an ellipse that can be represented by $E(a, b, x_0, y_0, \theta) = 0$ [6]. Here $2a$ and $2b$ are the lengths of the major and minor axes of the ellipse, with a, b drawn randomly from [4,16]. The center of the ellipse is (x_0, y_0), where x_0 and y_0 are randomly chosen from $[a, 32 - a]$. θ, the angle between the major axis of the ellipse and the x-axis, is a random number between 0 and π.

B. Simulation Results

In this subsection, we report our simulation results, each representing an averaged summary over 100 runs. The performance metrics include the detection accuracy and false alarm rate for faulty sensor detection, as defined by equation (4) in Subsection VI-A for the evaluation of $\mathcal{C}_1$, and the degree of fitting and false detection rate for event boundary detection, as defined by equations (5), (6), and (7) in Subsection VI-C for the evaluation of $\mathcal{C}_3$. Note that the two sets, $\mathcal{C}_1$ and $\mathcal{C}_3$, contain the detected faulty sensors and boundary sensors, respectively.

Figs. 9 and 10 plot the detection accuracy and false alarm rate *vs.* p, the probability that a sensor reading becomes faulty, under different network densities for faulty sensor detection. In Fig. 9, it is observed that the higher the p, the lower the detection accuracy. From Fig. 10, we observe that the false alarm rate decreases with p when density equals 20, 30, 40 and 50. By carefully tracing back the intermediate results, we find that the current settings of thresholds help to decrease the false alarm rate. From both graphs, we also observe that a higher network density often leads to a higher detection accuracy and 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 ≥ 30, the detection accuracy is above 94% and false alarm rate is around 1%. For density $= 20$, we achieve a detection accuracy of around 90% and false alarm rate $< 2\%$. From both graphs, we observe that a smaller number of sensors in $\mathcal{N}$ with density $= 10$ may not be a good choice for Algorithm 1.

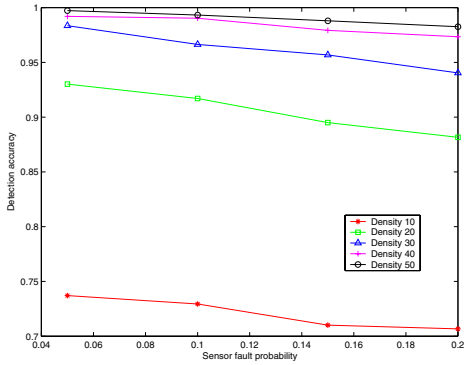


Fig. 9. Faulty sensor detection accuracy *vs.* p.

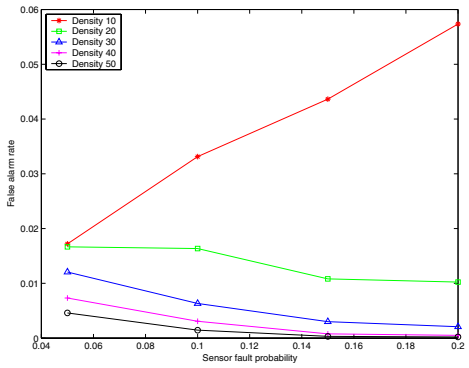


Fig. 10. False alarm rate in faulty sensor detection *vs.* p.

For event boundary detection described in Section V, we report the degree of fitting and the false detection rate *vs.* network density and *vs.* p, respectively, in Figs. 11-14. Note that in transition from $\mathcal{C}_1$ and $\mathcal{C}_2$ to $\mathcal{C}_3$, we need to set m, defined in Section V. Through simulation studies we observe that a larger m usually results in a higher

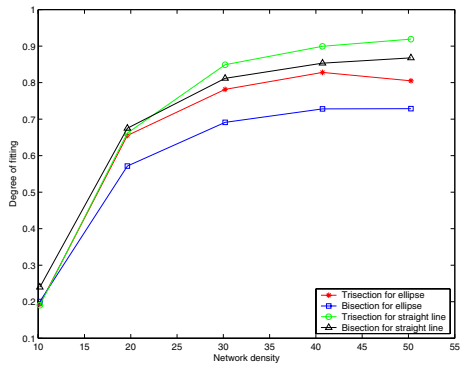


Fig. 11. Degree of fitting *vs*. network density when $p = 0$.

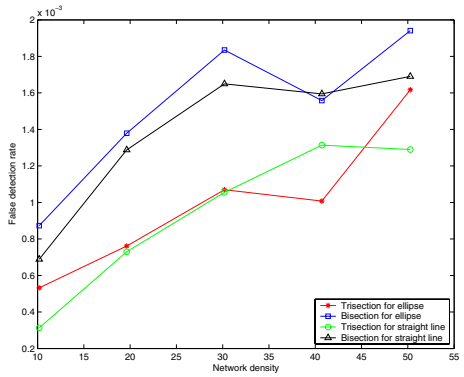


Fig. 12. False detection rate *vs*. network density when $p = 0$.

degree of fitting and a higher false detection rate, when the network density and p are fixed. In the following we fix $m = 4$ since this setting in general achieves a good degree of fitting and a low false detection rate.

A general observation from Figs. 11-14 is that for both linear and elliptical boundaries, the random trisection method outperforms the random bisection method. This is because random trisection induces a larger $\mathcal{NN}$. Besides, the degree of fitting for the linear boundary is higher than that of the elliptical curve. Now we examine each pair of figures in detail.

Figs. 11 and 12 demonstrate the performance of $\mathcal{C}_3$ with variable densities when $p = 0$. We observe that increasing density in general can increase both the degree of fitting and the false detection rate. This increase can be explained as follows. When we increase the density, i.e., the number of sensors in $\mathcal{N}$ and $\mathcal{N}^*$, we actually increase the transmission range R in our simulation. Thus the tolerance radius ($\frac{R}{2}$) is increased, which means more sensors close to the boundary will be assigned to $\mathcal{C}_2$. Then the size of $\mathcal{C}_3$ will be increased and thus the degree of fitting and the false detection rate will be increased accordingly. From the figures, we see that when the density is large enough (density ≥ 30), both the degree of fitting and the false detection rate increase slightly. This indicates that a density larger than 30 might affect the degree of fitting and false detection rate in about the same way as a density of 30.

We also observe several deviations from the "increasing trend" in Figs. 11 and 12. This is mainly due to the limited size of the simulation region, which becomes too small (compared with πR^2) when we increase the transmission range. Note that the unit of the y-axis also amplifies the deviation in Fig. 12.

When the sensor fault probability increases, the performance of $\mathcal{C}_3$ usually decreases, as shown in Figs. 13 and 14. Compared with Fig. 11, the degree of fitting is low when $p > 0$ for the same network density. This

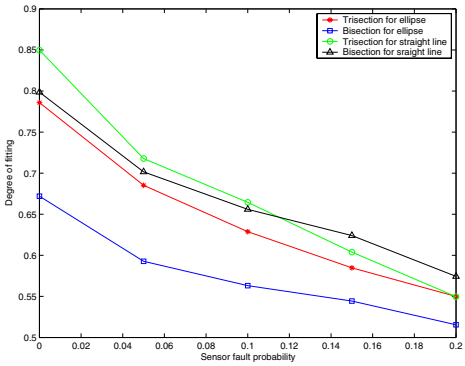


Fig. 13. Degree of fitting *vs*. p when density $= 30$.

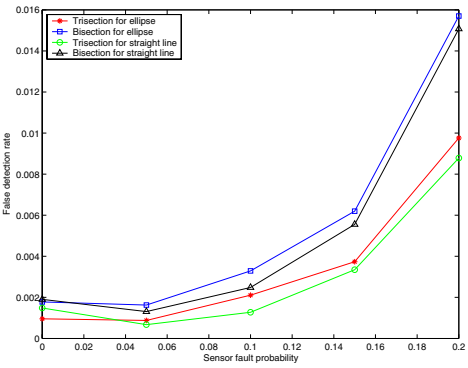


Fig. 14. False detection rate *vs*. p when density $= 30$.

is obvious since faulty sensors interfere with boundary nodes. Note that a low degree of fitting does not mean that the boundary can not be detected. It means that more sensors close to the boundary escape the detection. As shown in Fig. 2(d), with degree of fitting as low as 55% when $p = 0.2$, the elliptical boundary is still clearly identified. In this scenario, the faulty sensor detection accuracy is 91%.

We have also conducted simulations when input data are binary decision predicates and obtained results close

to those reported in Figs. 13 and 14. This indicates that our algorithms are applicable to both 0/1 decision predicates and numeric sensor readings.

VIII. Future Work Discussion

We believe that our ideas in detecting faulty sensors and event boundaries can be extended to multi-modality sensor networks, and the data aggregation can be done along both temporal and spatial dimensions for decreasing the false alarm rate in faulty sensor detection and the false detection rate for boundary detection. Thus we propose to explore along these directions in the future. The current algorithms are sensitive to the settings of thresholds, which are dependent on the sensor fault probability. By exploiting the false alarm probability obtained from base stations, an adaptive threshold that better fits the application context can be designed. We also target this as a future work. Besides, in our future work, we will explore the technique of curve fitting, instead of the metrics of degree of fitting and false detection rates, in order to evaluate the performance of our algorithm in detecting the event boundary.

Acknowledgment

D. Chen was supported by the National Science Foundation grant CCR-0311252.

The research of Dr. Xiuzhen Cheng is supported by NSF CAREER Award No. CNS-0347674.

References

[1] V. Barnet and T. Lewis, *Outliers in Statistical Data*, John Wiley and Sons, Inc., 1994.

[2] D. Chen and X. Cheng, An Asymptotic Analysis of Some Expert Fusion Methods, *Pattern Recognition Letters*, 22, 901-904, 2001.

[3] D. Chen, X. Cheng, and M. Ding, Localized Event Detection in Sensor Networks, *manuscript*, 2004.

[4] X. Cheng, D.-Z. Du, L. Wang and B. Xu, Relay Sensor Placement in Wireless Sensor Networks, to appear in *ACM/Kluwer Wireless Networks*, 2004.

[5] X. Cheng, A. Thaeler, G. Xue, and D. Chen, TPS: A Time-Based Positioning Scheme for Outdoor Sensor Networks, *IEEE INFOCOM*, HongKong China, March 2004.

[6] K.K. Chintalapudi and R. Govindan, Localized Edge Detection in Sensor Fields, *IEEE Ad Hoc Networks Journal*, pp. 59-70, 2003.

[7] T. Clouqueur, K.K. Saluja, and P. Ramanathan, Fault Tolerance in Collaborative Sensor Networks for Target Detection, *IEEE Transactions on Computers*, pp. 320-333, Vol. 53, No. 3, March 2004.

[8] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, Instrumenting the World with Wireless Sensor Networks, *ICASSP'01*, Salt Lake City, UT, 2001.

[9] J. Hill, A software architecture to support network sensors, Master's Thesis, UC Berkeley, 2000.

[10] J. Hill, R. Szewczyk, A. Woo, S. Hollar, and J. Heidemann, System architecture directions for networked sensors, *Proc. 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, November 2000.

[11] B. Krishnamachari and S. Iyengar, Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks, *IEEE Transactions on Computers*, Vol. 53, No. 3, pp. 241-250, March 2004.

[12] K. Krivoruchko, C. Gotway, and A. Zhigimont, Statistical Tools for Regional Data Analysis Using GIS, *GIS'03*, pp. 41-48, New Orleans, LA, November 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, March 2002.

[14] C.T. Lu, D. Chen, and Y. Kou, Detecting Spatial Outliers with Multiple Attributes, *Proceedings of 15th International Conference on Tools with Artificial Intelligence* pp. 122-128, November 2003.

[15] C.T. Lu, D. Chen, and Y. Kou, Algorithms for Spatial Outlier Detection, *Proceedings of 3rd IEEE International Conference on Data Mining*, pp. 597-600, November 2003.

[16] S. Madden, M. J. Franklin and J.M. Hellerstein and W. Hong, TAG: a tiny aggregation service for ad-hoc sensor networks, *OSDI*, December 2002.

[17] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, Wireless Sensor Networks for Habitat Monitoring, *ACM WSNA'02*, Atlanta GA, September 2002.

[18] S. Marti, T.J. Giuli, K. Lai, and M. Baker, Mitigating Routing Misbehavior in Mobile Ad Hoc Networks, *ACM MOBICOM'00*, pp. 255-265, Boston, MA, August 2000.

[19] T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos, Distributed Deviation Detection in Sensor Networks, *SIGMOD Record*, Vol. 32, No. 4, pp. 77-82, December 2003.

[20] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar, Spins: Security Protocols for Sensor Networks, *Wireless Networks*, Vol. 8, No. 5, pp.521-534, September 2002.

[21] W. Ruml, Y. Shang, and Y. Zhang, Location from Mere Connectivity, *Proceedings of the 4th ACM International Symposium on Mobile Ad hoc Networking and Computing* (MobiHOC03), pp. 201-212, 2003.

[22] J. Staddon, D. Balfanz, and G. Durfee, Efficient Tracing of Failed Nodes in Sensor Networks, *ACM WSNA'02*, pp. 122-130, Atlanta, GA, September 2002.

[23] S. Tanachaiwiwat, P. Dave, R. Bhindwale, and A. Helmy, Secure Locations: Routing on Trust and Isolating Compromised Sensors in Location-Aware Sensor Networks, *ACM SenSys'03*, pp. 324-325, Los Angeles, California, 2003.

[24] A. Thaeler, M. Ding, X. Cheng, and D. Chen, iTPS: An Improved Location Discovery Scheme for Sensor Networks with Long Range Beacons, to appear in Special Issue on *Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks* of *Journal of Parallel and Distributed Computing*, Fall 2004.

[25] S. S. Wilks, *Mathematical Statistics*, Wiley, New York, 1962.

[26] H. Wu, C. Wang, and N.-F. Tzeng, Novel Self-Configurable Positioning Technique for Multi-hop Wireless Networks, to appear in *IEEE/ACM Transactions on Networking*.

[27] N. Xu, A Survey of Sensor Network Applications, http://enl.usc.edu/~ningxu/papers/survey.pdf

[28] Y. Zhao, R. Govindan, and D. Estrin, Residual Energy Scans for Monitoring Wireless Sensor Networks, *IEEE WCNC'02*, pp. 78-89, Florida, March 2002.