Fault Tolerant Event Boundary Detection and Target Tracking in Sensor Networks

by Min Ding

B.E. June 1995, Tianjin University, China

M.S. June 1999, Institute of Science and Technology Information of China

A dissertation submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial fulfillment of the requirements
for the degree of Doctor of Science in Computer Science

January 31, 2009

Dissertation directed by

Xiuzhen Cheng

Associate Professor of Computer Science
The School of Engineering and Applied Sciences of The George Washington University
certifies that Min Ding has passed the Final Examination for the degree of Doctor of
Science as of November 18th, 2008. This is the final and approved form of the
dissertation.

Fault Tolerant Event Boundary Detection and Target Tracking
in Sensor Networks

Min Ding

Dissertation Research Committee:

Xiuzhen Cheng, Associate Professor of Computer Science, Dissertation Director

Hyeong-Ah Choi, Professor of Computer Science, Committee Chair

Simon Y. Berkovich, Professor of Computer Science, Committee Member

Nan Zhang, Assistant Professor of Computer Science, Committee Member

Songqing Chen, Assistant Professor of Computer Science of George Mason Univer-
sity, Committee Member
Acknowledgements

I wish to express my sincere gratitude to my advisor, Professor Xiuzhen Cheng, for her invaluable guidance on research, her financial support, and her encouragement and friendship.

I am grateful to my committee members, Professors Hyeong-Ah Choi, Simon Y. Berkovich, Nan Zhang and Songqing Chen for their efforts and contributions. I appreciate Prof Dechang Chen in Uniformed Services University of the health Sciences, who gave me fruitful discussions and wonderful suggestions. I also thank Prof. Andreas Terzis in Johns Hopkins University and Dr. I-Jeng Wang at Johns Hopkins University Applied Physics Laboratory for hosting me as an intern at JHU in 2006 spring and summer. My thanks also goes to Prof. Jiang Li in Howard University.

I had a pleasant stay at Mobile Ad-hoc and Sensor Networking Laboratory (MASNET) and Computer Science Department, the George Washington University. I am indebted to my friends at GWU and lab colleagues: Dr Fang Liu, Dr Liran Ma, and Kai Xing. I really enjoyed my collaborations and discussions with them.

Finally, I thank my son, my husband, and my parents, for their love and support during my D. Sc study.
Abstract of Dissertation

In the last decade, sensor network has been emerging as an indispensable application in the area of biological observation [31, 72], security surveillance [33, 59], traffic monitoring [5, 13], earth activity recording [71, 74] and others. Detecting event frontline or boundary sensors and tracking dynamically moving events in a complex sensor network environment are critical problems for sensor network applications. By considering the nature of sensor data, general data mining techniques are not directly applicable, which motivates us to investigate collaborative, distributed data mining methods that enables efficient distributed computation by individual sensor nodes with limited computation power and memory storage. In this thesis, we provide two classes of distributed in-network processing schemes, based on different task requirements, for outlier sensor detection, fault-tolerant event boundary detection and target tracking in sensor networks.

We first propose robust Median estimator based approaches for identification of outlying sensors and detection of the reach of events in sensor networks. To identify outlying sensors, median is used as the estimation of the observation in a close proximity. Accordingly, an outlier is detected by collaborative in-network comparison in the close proximity. As for event frontline detection, a special proximity is chosen such that the measurements of a sensor node close to the real event boundary significantly differentiate with the local sensing estimation in this special neighborhood.

We next introduce our exploration of using statistical clustering methods with model selection analysis [1, 2, 28, 67] for distributional sensor data modeling and event frontline sensor detection [25]. A Boundary sensor is considered as being associated with a multimodal local neighborhood of (univariate or multivariate) sensing readings, and each Non-Boundary sensor is treated as being with a unimodal sensor reading neighborhood. Furthermore, the set of sensor readings within each sensor’s spatial neighborhood is formulated using Gaussian Mixture Model. Two classes of Boundary and Non-Boundary
sensors can be effectively classified using the model selection techniques for finite mixture models. We further propose its temporally adaptive version for dynamic target tracking in changing environments, under a unified statistical mixture modeling framework. The proposed algorithms can be implemented within each purely localized sensor neighborhood and scale well to large-range sensor networks. The computational complexity is moderate and comparable to our previous Median based approaches. Our extensive experimental results demonstrate that our algorithms effectively detect the event boundary with a high accuracy under moderate noise levels. Desirable quantitative target tracking results are also achieved under challenging background conditions.
Contents

Acknowledgements iii

Abstract iv

Table of Contents vi

List of Figures xi

List of Tables xv

1 Introduction 1

1.1 What Is Sensor Network? 1

1.2 In-network Processing and Sensor Network 2

1.3 Fault-tolerant Event Boundary Detection and Target Tracking in Sensor Networks 3

2 Related work 8

2.1 Event Boundary Detection 8

2.2 Outlier Detection 10
## 3 Fault-Tolerant Event Boundary Detection in Sensor Networks based on Median Estimator

### 3.1 Introduction

### 3.2 Notations and Network Model

### 3.3 Localized Outlying Sensor Detection

#### 3.3.1 Derivation of Detection Procedure

#### 3.3.2 Algorithm

### 3.4 Localized Event Boundary Detection

### 3.5 Performance Evaluation

#### 3.5.1 Evaluation of $C_1$

#### 3.5.2 When Could Sensors be Assigned to $C_2$?

#### 3.5.3 Evaluation of $C_3$

### 3.6 Simulation

#### 3.6.1 Simulation Set-Up

#### 3.6.2 Determination of Thresholds

#### 3.6.3 Simulation Results
4 Event Boundary Detection in Sensor Networks based on Gaussian mixture model 46

4.1 Introduction .................................................. 46

4.2 Previous Work .................................................. 48
  4.2.1 Median based Event Boundary Sensor Detection [22, 76] .......... 48
  4.2.2 Analysis .................................................... 50

4.3 Motivation and Mixture Models .................................. 55
  4.3.1 Finite Mixture Models and Gaussian Mixture Models ............. 55
  4.3.2 Expectation Maximization Algorithm ................................ 56
  4.3.3 Model Selection ............................................. 58
  4.3.4 Initialization and Convergence: ................................ 60

4.4 Algorithms .................................................... 60
  4.4.1 Sensor Classification using Kmeans Clustering and Covariance Deduction .................................................... 61
  4.4.2 Sensor Classification based on Gaussian Mixture Model with Model Selection .................................................... 64

4.5 Experiments ................................................... 66
  4.5.1 Experiment Setup ............................................. 66
5 Target Tracking in Sensor Networks based on Gaussian mixture model 77

5.1 Introduction ................................................. 77

5.2 Algorithms .................................................. 78

5.2.1 Model Initialization for Target Appearance Distribution . 79

5.2.2 Mean-shift Optimization for Target Localization .......... 80

5.2.3 Adaptive GMMs for Target Updating ...................... 82

5.3 Simulations .................................................. 86

5.3.1 Simulation Setup ....................................... 86

5.3.2 Target Tracking Results ................................. 89

5.4 Discussions ............................................... 95

6 Conclusion and Future Work .................................. 98

Bibliography .................................................... 101

A Effectiveness constraint based Normality Property .......... 110
B  Farthest-point Clustering Algorithm for Initialization of $EM$  111

C  Kernel Mean-shift Target Tracking  112
List of Figures

3.1 \( \mathcal{N}^{*} \) 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. .................................................. 19

3.2 Illustration of \( C_1, C_2, \) and \( C_3 \). Data in (a) are obtained from one run of the experiment leading to Figs 1.12. 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 \( \circ \) in (b), (c), and (d) represents a node in \( C_1, C_2, \) and \( C_3 \), respectively. ........................................... 22

3.3 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. .................................................. 24

3.4 Illustration of random bisection. \( \mathcal{N}\mathcal{N}(S_i) \) is the half disk containing \( P_1, P_2, B, \) and \( S_i \). ................................................................. 25

3.5 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 \( 2\pi/3 \). \( \mathcal{N}\mathcal{N}(S_i) \) is the union of sectors i and iii. .............................. 26

3.6 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. 29

3.7 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. 30
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.8</td>
<td>An illustration of the square $Q$ fitted into the boundary area.</td>
</tr>
<tr>
<td>3.9</td>
<td>Examples of ROC curves for density = 30 and different values of $p$.</td>
</tr>
<tr>
<td>3.10</td>
<td>Detection accuracy $\text{vs.}$ sensor outlying probability $p$ for different density values.</td>
</tr>
<tr>
<td>3.11</td>
<td>False alarm rate $\text{vs.}$ sensor outlying probability $p$ for different density values.</td>
</tr>
<tr>
<td>3.12</td>
<td>Degree of fitting $\text{vs.}$ sensor outlying probability $p$ when density = 30.</td>
</tr>
<tr>
<td>3.13</td>
<td>False detection rate $\text{vs.}$ sensor outlying probability $p$ when density = 30.</td>
</tr>
<tr>
<td>3.14</td>
<td>Performance of the event boundary detection algorithm under different values of sensor outlying probability $p$ and network density (part-1).</td>
</tr>
<tr>
<td>3.15</td>
<td>Performance of the event boundary detection algorithm under different values of sensor outlying probability $p$ and network density (part-2).</td>
</tr>
<tr>
<td>4.1</td>
<td>Relationship of a local sensor neighborhood $\mathcal{N}(S_i)$ and the global neighborhood $\mathcal{N}^\ast$. $\mathcal{N}(S_i)$ is the local neighborhood of sensor $S_i$. If $S_i$ is lying at the event boundary, there are two Normal distributions of sensor readings existing in $\mathcal{N}(S_i)$. Otherwise, one single normal distribution can well present all sensor values in $\mathcal{N}(S_i)$. The global neighborhood $\mathcal{N}^\ast$ is defined to be a large enough area within one single event region which includes sufficient number of $\mathcal{N}(S_i)$ for their $\hat{\mu}$ and $\hat{\sigma}$ statistics estimation of one single Normal distribution. Blue sensors are within $\mathcal{N}^\ast$ to be considered as one event, red sensors are lying outside of $\mathcal{N}^\ast$ and from another sensing event. $\mathcal{N}(S_i)$, containing both red and blue sensors, is treated as event boundary sensor.</td>
</tr>
</tbody>
</table>
4.2 Sensor map represented using an image. (a) a local sensor map of two sensing events shown in different colors with their spatial locations, (b) a global sensor map represented by an image. 67

4.3 Examples of event Boundary and Non-Boundary sensor classification using Algorithm 4 under different event configurations. Rows (a.1-3), (b.1-3), (c.1-3) are for $EL_{OL}, LB_{OL}$ and star graph event shapes respectively. The (a-c.1) column shows the maps of simulated multivariate Gaussian samples as sensor readings; (a-c.2) column describes the ground truth boundary sensors shown with higher intensities; (a-c.3) column is the according classification results using algorithm 4. 71

4.4 Illustrative examples of the bi-modal performance phenomenon of algorithm 4 when $\lambda = 4.0 \times \phi$ in (a) or $\lambda = 2.5 \times \phi$ in (b). 76

5.1 A scenario for target tracking where the white circles represent sleeping nodes, blue circles denote active nodes, and red circles are event sensors that contribute to the target location estimation at each time frame. (a) shows the initial step for model fitting at frame $t-1$, (b) is the process for mean-shift optimization to estimate $\hat{R}(t)$ at each iterations, and (c) shows the final estimated target location at frame $t$. 83

5.2 An example of tracking a moving target over a trajectory trespassing the two background event regions in the sensor map. Red shows the ground truth (the trajectory of the target), while blue is the tracking trajectory from Algorithm 5 with an adaptive $GMM$ model; and green is associated with the fixed $GMM$ variant. It is obvious that Algorithm 5 can track the moving target over the entire spatial range while the fixed $GMM$ variant fails to track the target after a number of time frames. 90
5.3 Sampled frames of tracking the moving target using our proposed Algorithm 5 in a 180 frame sequence. Target is successfully tracked at both the non-event background area and the two event background regions. The red ellipse shows the target boundary based on the ground truth; the blue ellipse shows the tracked target boundary using Algorithm 5. ... 92

5.4 Sampled frames of tracking the moving target using the fixed GMM variant of Algorithm 5 in a 180 time frame sequence. Note that the fixed GMM variant loses the target at frame 19#. ... 93

5.5 Sampled frames of tracking the moving target using algorithm 5-B when 20% outlying sensors exist. The target is successfully tracked in a 180 frame sequence. ... 93

5.6 Plot of mean location errors of Algorithms 5-A and 5-B, based on the average of 100 trails per configuration. ... 96
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Relationship between $\theta_1$ and $p$ when $y$’s are iid from $N(0, 1)$.</td>
<td>35</td>
</tr>
<tr>
<td>3.2</td>
<td>False alarm rates and detection accuracies associated with the points indicated on the ROC curves in Fig. 3.9.</td>
<td>39</td>
</tr>
<tr>
<td>3.3</td>
<td>Angles of line segments of ROC curves in Fig. 3.9.</td>
<td>39</td>
</tr>
<tr>
<td>3.4</td>
<td>Optimal threshold values derived from Table 3.3.</td>
<td>39</td>
</tr>
<tr>
<td>4.1</td>
<td>Comparison of Median based method [22, 76], algorithm 3 and algorithm 4 under five Event Shapes.</td>
<td>72</td>
</tr>
<tr>
<td>4.2</td>
<td>Robustness evaluation on the outlying sensor ratios $\varpi$ of algorithm 4 in $EL_{OL}$.</td>
<td>73</td>
</tr>
<tr>
<td>4.3</td>
<td>Performance evaluation on the different $\zeta_1$ settings of algorithm 4 in $EL_{OL}$.</td>
<td>74</td>
</tr>
<tr>
<td>4.4</td>
<td>Performance sensitivity testing on covariance scales $\sigma$ of algorithm 4 in $EL_{OL}$.</td>
<td>74</td>
</tr>
<tr>
<td>4.5</td>
<td>Performance sensitivity testing on sensor neighborhood sizes $\phi$ of algorithm 4 in $EL_{OL}$.</td>
<td>75</td>
</tr>
<tr>
<td>5.1</td>
<td>Comparison study of Algorithm 5 and its fixed GMM variant under the general background noise model and the complex noise model. The first four lines of numerical results are for the Gaussian mixture target model, and the last four lines are computed using the spatially decaying target model.</td>
<td>94</td>
</tr>
</tbody>
</table>
5.2 Comparison of Algorithms 5-A and 5-B, using the criterion of rates of success of tracking versus total trials.
Chapter 1: Introduction

Sensor networking [8, 13, 14, 27, 29, 73] and information processing methods [6, 18, 19, 26, 35] (for sensing data) have emerged as an important and rapidly growing area of many research activities [10, 22, 25, 43, 49, 71, 83, 84] and real world applications, such as indoor and outdoor surveillance [29, 33, 66, 73, 79], hazard prevention as landslide prediction [65, 71] and volcano monitoring [74], biological and ecological system sensor networks [50, 55, 61, 72], traffic monitoring [5], and so on. Sensor network can provide a global scale sensing map through the collaboration of many sensors, where each observing a local view [43, 49]. With their capabilities for distributed sensing and in-network processing, networked sensors are expected to be widely deployed and to perform some decentralized information processing tasks such as environmental monitoring, disaster recovery and urban rescue, target identification and tracking. The difficulty in flourishing these networks lies in in-network processing observations from sensors in close geographic proximity [27]. The in-network processing in sensor network is a broad topic, mostly involving collaborative data integration and aggregation. In this dissertation, we mainly focus on proposing and validating two distributed in-network processing schemes [22, 23, 25], to address different task requirements for outlying sensor identification, fault-tolerant event frontline detection and target tracking and localization in sensor networks.

1.1 What Is Sensor Network?

A typical sensor network consists of a larger number of cooperating sensor nodes and one or more base stations that work as the fusion centers to collect data from sensors and make the final decision based on spatially or temporally aggregated data stream. Then the base stations are further connected with external components (i.e., Internet) and transform the information from the sensors to the end-users of the sensor network.
A sensor node is composed of a microprocessor and limited amount of memory and storage space with wireless communication capacity. Thus each node can directly communicate with a small number of neighboring nodes within the radio range. Both the computation capability and communication capability are limited. The sensor unit is also equipped with some sensing devices, such as magnetic [24], humidity, or light sensors [54]. The sensor nodes are powered by batteries which may not be recharged or replaced after deployment.

1.2 In-network Processing and Sensor Network

In the general field of information processing in sensor networks, we are facing three main challenges. First, the scale of a sensor network can be tremendous while often it may contain thousands of sensors deployed across a broad geographical region. Consequently, the volume of unordered in-network sensor readings over spatial and temporal domains is huge. For example, the surveillance purposed sensor network in Vigilnet [29, 33, 73] is designed to scale towards at least 1000 sensor motes and monitor a minimum of $1000 \times 1000$ square meters region. In [72], the sensor network consists of 33 motes for the biology ecological study of a redwood tree, which produces 1.7 million data samples within 44 days. Second, sensor readings are prone to noise in real environments and unreliable inter-sensor communications can cause further information loss. In the experiments of soil monitoring in Bangladesh [60, 61], the researchers collected approximately 25,000 data points in a period of 12 days and noticed that 24% of them are uninterpretable. Third, each sensor has very limited computational power, memory storage and energy supply, which make computation expensive algorithms less valuable. In-Situ data processing and mining is highly desirable.

These challenges motivate us to design robust, efficient, distributed and in-network information extraction algorithms [22, 23, 25]. Through in-network processing, the data is pre-processed before they are transmitted, thus redundant, useless and spurious data are
deleted, and partial observations from different sensors are combined and aggregated. The scalability and lifetime of sensor networks can be significantly improved (and enhanced by energy-aware routing design [21]). As sensor network may have a wide range number of sensor units, the proposed algorithms must be computational efficient in terms of network scalability as well.

How to design efficient in-network processing algorithms for sensor networks has attracted intensive research efforts. In-network aggregation is proposed to combine and aggregate the data from different sensors [21, 49, 78]. In-network aggregation can decrease energy and bandwidth consumption by reducing the data volume transformed to the base station, and extend the lifetime of the sensor network significantly. Collaborative signal and information processing (CSIP) [43, 84] is presented for collaboration between nodes to process the space-time signal in sensor network. CSIP algorithms have been successfully applied for target detection, classification and tracking [43, 45, 84]. In this thesis, we describe two distributed in-network processing schemes with respect to applications of outlying sensor identification, fault-tolerant event frontline detection and target tracking and localization in sensor networks [22, 23, 25]. Our work is attempting to bridge the intensively distributed sensing data to meet the information query needs of end-users. By considering the nature of sensor data, general data mining techniques are not directly applicable, thus we propose collaborative, distributed data mining methods that enables efficient distributed computation by individual sensor nodes with limited computation power and memory storage.

1.3 Fault-tolerant Event Boundary Detection and Target Tracking in Sensor Networks

As studied in [60, 61, 68], sensors are error-prone due to both possible hardware defects and large environmental variations. Under such conditions, these sensors will report
outlying values which may cause false alarm for the typical event detection applications. Consequently, one important task is to monitor, detect, and report the occurrences of interesting events with the presence of outlying sensor measurements, by leveraging robust in-network information processing methods. Furthermore, 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. Typical applications include the detection of the transportation front line of some vegetation or animalcule’s growth over a certain geographical region.

In this thesis, we mainly target on the identification of outlier sensors and detection of the reach of events in sensor networks with co-existing outlying sensors [22, 23, 25]. Our research is also motivated by the fact that filtering out both outlying readings and redundant data, and transmitting only the boundary information to the base station can conserve energy to a great degree. We noticed that other researchers also adopt similar strategies in real sensor network applications. For example, in volcano monitoring sensor network [74], sensor nodes are designed to detect interesting seismic events locally and transmit only event reports to the base station. The work in [31] focuses on misclassification of real events with outliers by employing some fault detection techniques [68].

We propose, analyze and validate two different mathematical schemes for outlying sensor identification and fault-tolerant event boundary detection, based on robust Median estimator [22, 23] and statistical Gaussian mixture models [25], respectively. Our first approach copes with sensor data of single modality. The basic idea of outlying sensor detection is given as follows: Each sensors first computes the difference between its own reading and the median value of the readings from its neighborhood sensor set. Each sensor then collects all differences from its neighborhood and standardizes them using variance normalization (ie. converting as Mahalanobis distances). A sensor is an outlier if the absolute value of its standardized difference is sufficiently large. The algorithm for event boundary detection 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 [22].

Our second method for outlying sensor identification and event boundary detection can handle multivariate data set under the condition of multiple channel sensor readings for event reporting, ie., statistical multivariate variables versus multi-modality sensor readings. For example, the MTS300 and MTS310 sensor boards [54] have a variety of sensing modalities, such as temperature, acoustic, and sounder, etc. In the application of this kind of sensors, more rich sensor information of different modalities can be combined and statistically fused to achieve better monitoring report. A principled ways of modeling multivariate dependency and correlations between the sensing data channels is in demand for many real applications [31, 60, 61, 72, 74]. In [60, 61], two types of sensors are deployed to collect temperature and moisture values, and seven additional soil chemistry reports are generated for soil monitoring. In [72], each data point collected for biological ecological analysis of redwood contains multivariate variables: temperature, humidity, incident PAR (as photosynthetically active solar radiation), and reflected PAR.

Our algorithms of event frontline detection [25] are derived from statistical Gaussian mixture models with explicit model selection schemes [1, 2, 28, 67]. The basic idea is based on the observation that a Boundary sensor is considered as residing within a local sensor neighborhood with a multimodal distribution of (univariate or multivariate) reading inputs, while each Non-Boundary sensor is surrounded in a neighborhood of unimodal sensing values. More precisely, the distribution of sensor readings within each sensor’s spatial neighborhood set is mathematically formulated using finite Gaussian Mixture Models (GMM) [28, 53]. To this end, model selection techniques [1, 2, 28, 67, 85] have been actively researched and developed in the past three decades for mixture models. They can be effectively utilized to identify the correct number of modes \( \Gamma \) for any Finite Mixture
Models [53], of which Gaussian mixture model (GMM) is the most popular one [28]. Thus Boundary and Non-Boundary sensors can be consequently distinguished based on the criterion whether its neighboring sensor data distributions having $\Gamma > 1$ or $\Gamma = 1$. Both of our robust Median and Gaussian mixture model based schemes [22, 23, 25] are purely localized and scale well to large sensor networks.

Another important type of application task for sensor network is to detect and track the locations of the moving targets (e.g. vehicles, land mines, etc.) [8, 73]. Due to the stingy energy budget within each sensor, it is not affordable to let all sensors that detect the target presence send the data to the base station. we have to seek localized and computationally efficient algorithms such that the sensors can locally determine whether a target presents and then locate the target position. Fault-tolerant target detection and tracking is a challenging task in collaborative sensor networks.

In our current work [25], we mainly focus on the detection and tracking of the occurrences of interesting events that spread in a geographical region. We extend our GMM based scheme to a temporally-adaptive version [52] for dynamic target appearance modeling and apply high-performance mode-seeking optimization [15, 16] to track a target moving in a geographical region with changing background sensing signal models. In this scheme, multivariate Gaussian density functions of mixture models are used to capture the statistical target properties and estimated using temporal updates. Meanwhile, mean-shift algorithm [15, 16, 48] is exploited to predict and converge the moving target locations over time. Our mixture model based algorithm is capable of fusing multivariate real-valued sensor measurements and its probability nature shows fault tolerance and robustness in noisy sensing environments.

As summary, we propose two algorithmic schemes [22, 23, 25], rooted from robust Median estimator [35] and Gaussian mixture models (GMM) with model selection [2, 19, 67] respectively, for outlying and boundary data detection and target tracking in sensor networks. The incrementally adaptive GMM [52, 56] is also employed to form an target
tracking algorithm executed in a dynamic geographical region [25]. Both classes of
schemes have low computation overhead, and are purely localized. The median based
scheme is much simpler, however only deals with single modality. The GMM based
method provides a principle way to modeling and fusing multivariate sensor readings.
To the best of our knowledge, our mixture model based approach is the first work that
presents a principled methodology integrating multi-modality sensor readings for event
boundary detection. For the rest parts of this thesis, we will present these two lightweight,
mathematical schemes [22, 23, 25] in details and report the corresponding experimental
analysis for outlying detection, fault-tolerant event boundary detection and target tracking
respectively. Comparisons between these two classes of schemes for event boundary
detection, and target tracking under different target appearance model settings (within
scheme 2) are also given.
Chapter 2: Related work

In this chapter, we briefly survey related results in sensor network research along three major lines:

- event boundary detection,
- outlier detection,
- and target tracking.

2.1 Event Boundary Detection

[14] is among the early work of localized boundary detection in sensor networks. The authors propose three different schemes as statistical approach, image processing approach, and classifier-based approach, all of which only takes inputs of 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 according to a threshold whose value can be obtained using a lookup table. The image processing approach instead computes a weighted average of all the neighboring sensor predicates. The classifier-based approach further train 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 Median based boundary detection algorithm [22] does not require the pre-computation of the 0/1 decision predicates. It is also computationally more efficient than the last two approaches in [14]. Furthermore the threshold in our algorithm only depends on the fault tolerance requirement.
[40] presents a noise-tolerated algorithm named *NED* for event and event boundary detection. In *NED*, the moving mean of the readings of the neighboring node set is used as the estimate for a certain sensor node. Then the event boundary is determined by trimming the estimated sensor readings around an event threshold based on a certain confidence level. Compare to the moving mean estimation, *Median* [22] is a more robust estimator of the “center” of random samples. This is because the sample mean can not represent effectively the “center” of samples when some values of the samples are extreme or outlying.

[37] proposes an event boundary detection method using autonomous agents which localize node transmission along the event boundary. Firstly, a node that detects an event within a preset time window will randomly generate an agent (parent agent) based on a preset threshold probability. Then, the agent moves along a random direction vector. A boundary can be determined once the agent hears a response from a node that has not seen the event. At the same time, a child agent is produced and transmitted along the opposite direction. Thirdly, both the parent and child agents sorts their event neighbors and moves to the nearest event node along their direction vectors. The final event boundary is determined and report to the base station when a parent agent meets a child agent. This scheme depends on the success transmission of the agents. It generates the event boundary sequentially and may cause some latency when the event region spans a large area. Meanwhile, the algorithm may fail once the agent packages lose. On the other hand, the outlying sensor readings may cause inaccurate boundary determination.

In [63], both security issue and event boundary detection application are considered. An efficient key establishment protocol is first proposed to secure the communications between neighboring nodes, and also between nodes and the sink. Then a collaborative endorsement scheme is designed to enhance fault-tolerant event boundary detection based on three statistical rules: *majority* rule, *consistency* rule, and *determination* rule [63]. A node keeps its own measurement only if this measurement is the majority among the neighboring readings. The *majority* rule helps to correct sensor measurements errors. Alternatively the
neighbors neglect such kind of measurements according to the *consistency* rule. In the *determination* rule, a boundary node is decided by comparing the event nodes and non-event nodes in the neighborhood in a statistical sense. The work in [63] considers both the security issue and fault-tolerance in event boundary detection. However, it only takes 0/1 input during the boundary determination step as [14].

To the best of our knowledge, our *Gaussian* Mixture Model based event boundary algorithm [25] is the first to process the multivariate sensor readings. It provides a principled way to model the covariance and cross-correlation between multivariate sensor measurements, which is capable to model the multi-modality sensing in a formal mathematical manner. The experimental performance evaluation also validates its superiority and improvement, versus our previous work using *Median* [22].

### 2.2 Outlier Detection

Recently, outlier sensor data detection has attracted interests in various real world applications, such as ground water analysis in Bangladesh [60,61], soil monitoring network at Jug Bay wetlands sanctuary [31], environmental monitoring in the redwoods in Sonoma California [72] and so on. Spatial outlier detection and regional data analysis have been extensively studied in spatial data mining research [42,46,47]. For the study of the ground water analysis [60,61], the researchers design a rule-based fault detection system to fill out the unreliable sensor reports in soil sensing. Different kinds of sensor and hardware faults are described based on the analysis of the collected data samples: large time gradients, stuck-at value, broken sensor, disconnected sensor, Low battery, and lighting. Upon the evaluation, five rules are proposed to detect the outlying sensor as *Noise, Invalid NLDR, Broken, Short*, and *Bad battery sensor*.

Sharma et al. [68] characterize three kinds of sensor faults: *short* fault, *noise* fault, and
constant fault, based on the evaluation of several real world sensor data sets. A “sharp” change between two successive data samples is identified as a short fault. In the analysis of sensor data-sets [60, 68], it shows a large increase in the variance of the successive sample readings when the battery of a sensor node is insufficient. Such kind of fault is defined as noise fault. As for a constant fault, a sensor node reports a constant abnormal value for a large number of successive data points. Accordingly, three fault detection methods are proposed to identify different kinds of sensor faults: Linear Least Squares estimation (LLSE) [35], Hidden Markov Models (HMM) [38], and Rule based method. In rule based method, the standard deviation of data samples within a predefined temporal window size is computed to detect the noise fault and constant fault, and the rate of change between two successive sample readings is used to identify the sharp fault. LLSE is a parameter estimation method which explores the spatial and temporal correlation within sensor readings in a dense sensor network. The covariance between measurements from different sensors is computed to identify the faulty readings based on a trained threshold. Furthermore Hidden Markov Model is a supervised learning based method, which is used to study and construct the anomalous pattern of the sensor readings (in time domain).

The authors in [31] implement both rule based and estimation based fault detection techniques [68] for soil monitoring, in the Jug Bay wetlands sanctuary. Their work is more focused on the study of mis-classification problem of the faults detection and event detection. As we have noted above, when a remarkable change in the readings of sensors is detected, an outlier or some event must have occurred. This observation is explored in [11, 43, 58] for 0/1 decision predicate computation. The related algorithms require only the most recent readings (within a sliding time window) of individual sensors. No collaboration among neighboring sensors is exploited (ie., in spatial domain). In [11], the “change point” of the time series are statistically computed based on Bayesian regression. The result can be used to answer questions such as “when will the front-line of the contamination reach a specified location?” The detector proposed in [43] computes a running average and compares it with a threshold, which is adjusted by a false alarm rate. In [58], 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 [22, 25] can discern outlying sensors from event sensors and compute the boundary of the event region, from a sound statistical basis.

2.3 Target Tracking

Target tracking has emerged as an interesting problem representing a very important class of sensor network applications [7,32,34,43,83]. In this section, we briefly survey the most related research.

Ref. [7, 43, 83] treat the sensor readings as multimodal feature vectors, where the mixture of multivariate Gaussian density functions or the mixture models are used to capture the statistical target properties. Based on either the probability density values or the (Mahanabolis) distances from the sensors influenced by the target, [7, 43, 83] train a k-Nearest Neighbor (KNN) or other types of classifiers to track the target by classifying the sensor readings in its predicted area at time $t$ from $t - 1$. In this thesis, we leverage on the adaptive Gaussian mixture model to capture the dynamic target appearance and adopt the continuous Mean-shift optimization for target localization over time. Our algorithm can successfully track the target moving trajectory under complex background noise scenarios. Compared with the pure nonparametric treatment of KNN type classifiers, our density evaluation is also more efficient.

A real-time target tracking system using wireless sensor network is designed and implemented in [32,34]. A power management protocol is adopted to set the sensor nodes either in active or in sleep states for prolonged network lifetime. In addition, some wake-up
strategy and group aggregation schemes are designed and analyzed for target detection, classification, and tracking tasks. The trade-off between energy saving and time delay in each phase is analyzed to guarantee the end-to-end tracking deadline in real applications. These works focus on the system design to achieve real-time target tracking in a timely and energy-efficient manner while our paper focuses on the algorithmic aspect of accurately tracking a moving target under simple and complex background noise models.

The authors in [80] also explore the trade-off between energy consumption and tracking quality in a networked sensor system. A quality-aware information collection protocol is proposed to determine which sensor is in active state by considering the tracking error tolerance of triangulation estimations. In [81, 82] a spanning tree rooted at the sensor node close to a target is used for target tracking, with the target position estimated by the location of the root sensor. In this paper, we propose more statistically-oriented algorithms for mobile target identification and localization, which allows us to directly model the distributional properties of sensor signals.

In [39], an R-tree sensor network topology is adopted for the detection and tracking of region-based targets. Two approaches: forward-all and forward-description methods are proposed for the detection of event regions. Furthermore, the authors describe three detailed algorithms: boundary detection algorithm, merging algorithm and description improvement algorithm to deal with on the problems on how to detect an event boundary, how to merge the event region obtained from the child nodes in the R-tree, and how to simplify and smooth the event boundary. Tracking of the event region is solved simply by computing the moving direction, moving speed, and expansion rate of the event, based on the the centers of the moving event at two time slots $t_1$ and $t_2$.

The authors in [44] propose a CLOUD framework to track the region-based event. The basic idea is to dynamically form a tree-based collective structure for each event region in each time slot. Both of their approaches in [39, 44] are limited by the dependence on the tree structure for the network topology.
The research work in [69] focuses on event detection and tracking in *Heterogeneous* sensor networks. First, the event region is defined as the overlapping between multiple attribute regions that are detected by different kinds of sensors. Furthermore, a distributed protocol, *CollECT*, is proposed for collaborative event detection among multiple sensing modalities. *CollECT* consists of three major procedures: vicinity triangulation for individual attribute region identification, event determination, and border sensor selection. The tracking is just implemented by performing the proposed procedure repeatedly. All the above work for region-based event tracking [39,44,69] did not address how to predict the event location for the tracking goal. Our *Mean-shift* powered optimization event tracking algorithm [25] can estimate the moving target’s location, in an adaptive-scaled gradient descend fashion [15], which not only guarantees algorithm convergence and usually generates desirable results.
Chapter 3: Fault-Tolerant Event Boundary Detection in Sensor Networks based on Median Estimator

3.1 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 animalcules 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 [41]. 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:

- The input can be any numeric value. This is significantly different from the existing work [14, 41] based on 0/1 decision predicates, where 1 indicates the occurrence of some phenomenon, and 0 indicates a normal status.

- The computation overhead is low, which involves only simple algebraic operations.

- The communication overhead is low, since sensor readings are disseminated to the neighborhood only.

- Event boundaries can be accurately identified even when many sensors report outlying measurements. In other words, outliers and boundary sensors can be clearly differentiated.
3.2 Notations and Network Model

Throughout this thesis, 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 $\mathbb{R}^2$. A sensor’s reading is outlying (or abnormal) if it deviates significantly from other readings of its neighborhood sensors [4]. Sensors with outlying readings are called outliers or outlying sensors, whereas sensors with normal readings are called normal sensor. In this thesis, the $i$th 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. Therefore, $x_i$ can be continuous or discrete. $x_i$ can also be extended from a continuous, univariate random variable to a multivariate one, as we discussed in chapter 4.

Informally, an event can be defined in terms of sensor readings. An event, denoted by $\mathcal{E}$, is a subset of $\mathbb{R}^2$ such that readings of the sensors in $\mathcal{E}$ are significantly different from those of sensors not in $\mathcal{E}$ (in a statistical manner). A sensor detecting some event is called an event sensor. An outlying sensor can be viewed as a special event which contains only one point, that is, the sensor itself. A point $x \in \mathbb{R}^2$ is said to be on the boundary of $\mathcal{E}$ if and only if each closed disk centered at $x$ contains both points in $\mathcal{E}$ and points not in $\mathcal{E}$. The boundary of the event $\mathcal{E}$, denoted by $B(\mathcal{E})$, 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. In chapter 4, the definition of event boundary sensor is further relaxed from geometrical sense, to distributional for better statistical stability. An event boundary sensor has its neighboring sensor reading set satisfied a multi-modal Normal distribution, rather than a uni-modal Gaussian distribution for any non-boundary sensor.

We assume that each sensor can locate its physical position through either GPS or GPS-
less techniques such as [12, 64, 70, 77]. Note that in this thesis we focus on the detection of outliers and event boundary sensors, and the estimated location of tracking event; thus report generation and delivery to the base station is out of scope, and will not be discussed. At last we assume that there exists a media access control (MAC) layer protocol to coordinate neighboring broadcastings so that no collision occurs.

3.3 Localized Outlying Sensor Detection

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

3.3.1 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 $\mathbb{R}^2$ that contains the sensor $S_i$ and additional $k$ sensors $S_{i1}, S_{i2}, \ldots, S_{ik}$. 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 $x_{i1}^{(i)}, x_{i2}^{(i)}, \ldots, x_{ik}^{(i)}$ denote the measurement at $S_{i1}, S_{i2}, \ldots, S_{ik}$, respectively. A comparison between $x_i$ and $\{x_{i1}^{(i)}, x_{i2}^{(i)}, \ldots, x_{ik}^{(i)}\}$ could be done by checking the difference between $x_i$ and the “center” of $\{x_{i1}^{(i)}, x_{i2}^{(i)}, \ldots, x_{ik}^{(i)}\}$. Clearly, such a difference is

$$d_i = x_i - \text{med}_i,$$  \hspace{1cm} (3.1)

where $\text{med}_i$ denotes the median of the set $\{x_{i1}^{(i)}, x_{i2}^{(i)}, \ldots, x_{ik}^{(i)}\}$. We note that $\text{med}_i$ in equation (3.1) should not be replaced by the mean $(x_{i1}^{(i)} + x_{i2}^{(i)} + \cdots + x_{ik}^{(i)})/k$ of the set $\{x_{i1}^{(i)}, x_{i2}^{(i)}, \ldots, x_{ik}^{(i)}\}$. 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 an outlier. 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 \( \mathcal{N}^*(S_i) \subset \mathbb{R}^2 \) 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 3.1 for an illustration of \( \mathcal{N} \) and \( \mathcal{N}^* \). According to equation (3.1) 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 an outlying 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.,

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

\[
\hat{\sigma} = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (d_i - \hat{\mu})^2}.
\]
Standardize the dataset $D$ to obtain \{\(y_1, \ldots, y_i, \ldots, y_n\}\}, where
\[
y_1 = \frac{d_1 - \hat{\mu}}{\hat{\sigma}}, \ldots, y_i = \frac{d_i - \hat{\mu}}{\hat{\sigma}}, \ldots, y_n = \frac{d_n - \hat{\mu}}{\hat{\sigma}}.
\] (3.2)

**DECISION**: If \(|y_i| \geq \theta\), treat \(S_i\) as an outlying 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 \(N(S_i)\), assume \(x^{(i)}_1, x^{(i)}_2, \ldots, x^{(i)}_k\) form a sample from a population having a continuous distribution function \(F\). Let \(x^{(i)}_1, x^{(i)}_2, \ldots, x^{(i)}_k\) be rearranged in the order from least to greatest and let the ordered values be \(x^{(i)}_1, x^{(i)}_2, \ldots, x^{(i)}_k\), where \(x^{(i)}_1 \leq x^{(i)}_2 \leq \cdots \leq x^{(i)}_k\). Then equation (3.1) can be rewritten as
\[
\text{med}_i = \begin{cases} 
 x^{(i)}_{((k+1)/2)} & \text{if } k \text{ is odd} \\
 \left(\frac{x^{(i)}_{(k/2)} + x^{(i)}_{(k/2+1)}}{2}\right) & \text{if } k \text{ is even.}
\end{cases}
\] (3.3)

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 \(k \to \infty\), \(\text{med}_i\) converges in probability to \(\tilde{\mu}\).

To prove, we first note the following special case of Theorem 9.6.5 in [75]: 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 \([a]\) denote the largest integer less than or equal to \(a\) and let \((a)\) denote the difference \(a - [a]\). Then \(0 \leq (a) < 1\). Set \(p_{1k} = \frac{[0.5k] + 1}{k}\). 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 \( x_{(k p_{1k})}^{(i)} \) and \( x_{(k p_{2k})}^{(i)} \) converge in probability to \( \bar{\mu} \). Now the proposition follows from the observation that equation (3.3) is equivalent to the following

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

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 [9].

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 outlying. When \( \theta = 2 \), the probability of observing a \( y_i \) with \( |y_i| \geq 2 \) is about 5%.

### 3.3.2 Algorithm

Let \( C_1 \) denote the set of sensors with \( |y_i| \geq \theta \). The set \( C_1 \) is viewed as a set of sensors that are claimed as outliers by the above procedure. The procedure in part 3.3.1 can be
Figure 3.2: Illustration of $C_1$, $C_2$, and $C_3$. Data in (a) are obtained from one run of the experiment leading to Figs 1.12. The interior of the ellipse is the event region. A sensor becomes outlying with probability $p = 0.2$. $\cdot$ represents a sensor and $+$ represents an outlying sensor. $\circ$ in (b), (c), and (d) represents a node in $C_1$, $C_2$, and $C_3$, respectively.

summarized into the following algorithm.
Algorithm 1 Median Based Outlying Sensor Detection

1. Construct \( \{N\} \) and \( \{N^*\} \). For each sensor \( S_i \), perform the following steps.
2. Use \( \{N(S_i)\} \) and equation (3.1) to compute \( d_i \) for sensor \( S_i \).
3. Use \( \{N^*(S_i)\} \) and equation (3.2) to compute \( y_i \) for sensor \( S_i \).
4. If \( |y_i| \geq \theta \), where \( \theta > 1 \) is predetermined, assign \( S_i \) to \( C_1 \). Otherwise, \( S_i \) is treated as a normal sensor.

Clearly, \( |C_1| \), the size of \( C_1 \), depends on \( \theta \). Assuming the \( y \) values in equation (3.2) constitute a sample of a standard normal distribution and the decisions are made independently, then if \( \theta \) is such that the right tail area of the density of \( N(0, 1) \) is \( \alpha \), \( |C_1| \) will be about \( \alpha \times N \).

In practice, a sensor becomes outlying 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 outlying sensors do not reflect reality, so that 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.

3.4 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. As mentioned above, \( C_1 \) 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 $\sigma$ is small compared to $|\mu_1 - \mu_2|$. \{N\} and \{N^*\} are constructed using closed disks. Consider sensor $S_i$. Assume that readings of sensors in a neighborhood of $S_i$ are within $2\sigma$ distance from the means of their corresponding normal distributions. Take each $N$ neighborhood of sensors in $N^*(S_i)$ to be sufficiently large. Due to uniformity of the deployment of sensors, calculation based on equation (3.1) shows that each $d$ follows $N(0, \sigma^2)$ approximately. For example, at sensor $S_1$ shown in Figure 3.3, $d_1$ follows approximately $N(0, \sigma^2)$. The reasoning is as follows. Let $R_1$ denote the portion of $N(S_1)$ that lies on the right hand side of the event boundary, and $R_2$ denote the remaining portion of $N(S_1)$. 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 $\text{med}_1$ will be obtained using the sensor readings from $R_1$. When $\sigma$ is small compared to $|\mu_1 - \mu_2|$, $\text{med}_1$ is about $\mu_2$, 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 (3.2) 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 $\theta = 2$. 

Figure 3.3: 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 $N^*(S_i)$. Both $N^*(S_i)$ and $N(S_1)$ are closed disks.
Figure 3.4: Illustration of random bisection. \( \mathcal{N}(S_i) \) is the half disk containing \( P_1, P_2, B, \) and \( S_i \).

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 \( \mathcal{N}(S_i) \) 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.

**Random Bisection**

Consider an \( S_i \) from the set \( S - C_1 \). 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 \( \mathcal{N}(S_i) \) to denote the half disk yielding the largest \( |d_i| \). For an illustration, consult Figure 3.4. 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 \( \mathcal{N}(S_i) \).

After \( \mathcal{N}(S_i) \) 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 (3.2) and make a decision on \( S_i \).

**Random Trisection**

Consider a closed disk centered at \( S_i \in S - C_1 \). Randomly divide the disk into three sectors with an equal area. Number the sectors as i, ii, and iii, as shown in Figure 3.5. 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 \( \mathcal{N}\mathcal{N}(S_i) \). It is easy to see that in Figure 3.5, \( \mathcal{N}\mathcal{N}(S_i) \) 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 \( \mathcal{N}\mathcal{N}(S_i) \). 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 \( B(E) \). Let \( C_2 \) denote the set of all the sensors with \( |y_i| \geq \theta \). The set \( C_2 \) is expected to contain enough sensors close to the event boundary (See Fig. 3.2(c)). In general, \( C_2 \) catches more sensors near the event boundary than \( C_1 \) does. Now we discuss how to combine \( C_1 \) and \( C_2 \) to infer the event boundary.
As seen in the derivation of $C_1$ and $C_2$, the set $C_1$ is expected to contain outlying sensors and $C_2$ is expected to contain sensors close to the event boundary. However, in general, $C_1$ also contains some sensors near the boundary, which are not outlying, and $C_2$ contains some sensors that are not close to the boundary. We now present a method to combine $C_1$ and $C_2$ 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 $C_1 \cup C_2$ to approximate the boundary. For a sensor $S_i \in C_1 \cup C_2$, 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 $C_2$ 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 $C_2$ that is different from $S_i$. For any positive integer $m$, let $C_3(m)$ denote the subset of $C_1 \cup C_2$ such that for each $S_i \in C_3(m)$, the disk $D(S_i; \sqrt{\frac{m b^2}{\pi N}})$ contains at least one sensor from $C_2$ that is different from $S_i$. The set $C_3(m)$ will serve as a set of sensors used to infer the event boundary (See Figure 3.2(d)). For convenience, sometimes we will write $C_3$ for $C_3(m)$.

Now we summarize the above procedure of finding $C_2$ and $C_3$ into the following algorithm.

**Algorithm 2 Median Based Event Boundary Sensor Detection**

1. Construct $\{N\}$ and $\{N^*\}$. Apply Algorithm 1 to produce the set $C_1 (\theta = \theta_1)$.

2. For each sensor $S_i \in S - C_1$, perform the following steps. Obtain $NN(S_i)$ and update $d_i$ from step 1) to the new $d_i$ from $NN(S_i)$, keeping unchanged all the other $d$ values from $N^*(S_i)$ obtained in step 1). Use equation (3.2) to recompute $y_i$. If $|y_i| \geq \theta$, assign $S_i$ to set $C_2 (\theta = \theta_2)$; otherwise, treat $S_i$ as a normal sensor.

3. Obtain $C_3(m)$, where $m$ is a predetermined positive integer.

We stress on the following points on the use of the 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 the $y$ values in equation (3.2) constitute a sample of a standard normal distribution and the decisions are made independently, then if $\theta_1$ and $\theta_2$ are such that the right tail areas of the density of $N(0, 1)$ are $\alpha_1$ and $\alpha_2$, respectively, the size of $C_2$ is about.
(1 − α_1) × α_2 × N. Third, unlike Algorithm 1, which utilizes the topological information of sensor locations to find C_1, Algorithm 2 uses the geographical information of locations to locate C_2 and C_3.

3.5 Performance Evaluation

Evaluation of the proposed algorithms include two tasks: evaluating C_1 and evaluating C_3. In this section, we first define metrics to evaluate C_1. Then we examine what type of sensors could be detected as being in C_2 by algorithm 2. The finding is then used to define metrics to evaluate the performance of C_3.

3.5.1 Evaluation of C_1

To evaluate the performance of C_1, we compute the detection accuracy $a(C_1)$, defined to be the ratio of the number of outlying sensors detected to the total number of outlying sensors, and the false alarm rate $e(C_1)$, 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

\[
a(C_1) = \frac{|C_1 \cap \mathcal{O}|}{|\mathcal{O}|}, \quad e(C_1) = \frac{|C_1 - \mathcal{O}|}{N - |\mathcal{O}|}.
\] (3.4)

If $a(C_1)$ is high and $e(C_1)$ is low, Algorithm 1 has a good performance.
Figure 3.6: 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.

### 3.5.2 When Could Sensors be Assigned to $C_2$?

Here we present a brief examination on what kind of sensors have the potential to be detected by algorithm 2 as belonging to $C_2$. 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 $\{N,N\}$.

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 $B(\mathcal{E})$ 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 $\mathcal{N}(S_i)$ will not become extreme among $d$ values from $\mathcal{N}(S_i)$ so that $S_i$ may not be detected as a sensor in $C_2$. Therefore, we only need to consider two cases shown in Figure 3.6 and Figure 3.7, where $B(\mathcal{E})$ intersects $D(S_i; R)$ in at least two sectors.

Consider Figure 3.6, where $B(\mathcal{E})$ intersects $D(S_i; R)$ in two sectors. Clearly, $\mathcal{N}(S_i)$ should be the union of sectors $i$ and $iii$. The event boundary cuts $\mathcal{N}(S_i)$ 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 $\angle AS_i B$. And consequently, the area of the other part is

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

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

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

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 Figure 3.7, where $B(\mathcal{E})$ intersects $D(S_i; R)$ in all three sectors. Let $\omega \in (0, \pi]$ be the value of $\angle AS_i B$. Then $\omega \geq 2\pi/3$. So $r = R \cos \frac{\omega}{2} \leq \frac{R}{3}$. Summarizing the above
shows that $r < \frac{R}{2}$.

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

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

### 3.5.3 Evaluation of $C_3$

Here we first describe a quantity to judge how well $C_3$ can be used to fit the boundary. Then we present a quantity to examine how many sensors that are “far away” are included in $C_3$. We begin with the following definition.

**DEFINITION:** For a positive number $r$, let $\text{BA}(\mathcal{E}; r)$ 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 $C_3$ is defined to be

$$a(C_3, r) = \frac{|\text{BA}(\mathcal{E}; r) \cap C_3|}{|\text{BA}(\mathcal{E}; r) \cap S|}.$$  \hfill (3.5)

Intuitively, $\text{BA}(\mathcal{E}; r)$ is a strip with width $2r$ centered around the event boundary. The quantity $a(C_3, r)$ 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 $\text{BA}(\mathcal{E}; r)$ is such that all the sensors in $\text{BA}(\mathcal{E}; r)$ provide a good outline of the boundary $B(\mathcal{E})$. If $a(C_3, r)$ is large, say, above 90%, then all the sensors in $\text{BA}(\mathcal{E}; r)$ 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 $\text{BA}(\mathcal{E}; r)$ and $a(C_3, r)$. If $r$ is large, say, above $R/2$, then the above Section 3.5.2 shows that many sensors within $\text{BA}(\mathcal{E}; r)$
will not be detected so that \( a(C_3, r) \) can be very low. On the other hand, if \( r \) is very small, \( \text{BA}(E; r) \) may become a strip containing few sensors so that \( \text{BA}(E; r) \) does not present a good description of the boundary \( B(E) \). A natural question is then: how can one choose an appropriate \( r \) such that \( \text{BA}(E; r) \) provides a good outline of the boundary and \( a(C_3, r) \) 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 \( \text{BA}(E; r) \), randomly draw a square \( Q \) “inside” \( \text{BA}(E; r) \) such that i) its width is \( 2r \); ii) two sides of the square are “perpendicular” to the boundary \( B(E) \). See Figure 3.8 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

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

For \( \text{BA}(E; r) \) 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
\[
r_1 = \frac{1}{2} \left( \frac{b}{\sqrt{N}} \right). \tag{3.6}
\]
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 \( C_3 \). Motivated by Section 3.5.2, we only check those sensors whose distances to the boundary are at least \( R/2 \). Let \( A(\mathcal{E}; R) \) denote the set of all points in \( \mathbb{R}^2 \) such that the distance of each point to the boundary \( B(\mathcal{E}) \) is at least \( R/2 \). Define the false detection rate of \( C_3 \) to be the following quantity
\[
e(C_3, R) = \frac{|A(\mathcal{E}; R) \cap C_3|}{|A(\mathcal{E}; R) \cap S|}. \tag{3.7}
\]
If \( e(C_3, R) \) is small, sensors far away from the event boundary are not likely to be contained in \( C_3 \).

### 3.6 Simulation

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

#### 3.6.1 Simulation Set-Up

We use MATLAB to perform all simulations. All the sensor nodes are uniformly distributed in a \( 64 \times 64 \) square region. The number of nodes is 4096.

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. “Event” sensor readings are drawn from $N(\mu_2, \sigma_2^2)$, comparing that Normal sensor readings are drawn from $N(\mu_1, \sigma_1^2)$. 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 $\sigma_1$ and $\sigma_2$. We choose $\sigma_1 = \sigma_2 = 1$ because they represent the system calibration error which should be small for a sensor that is not outlying.

In all the simulation scenarios, we choose $\mathcal{N} = \mathcal{N}^*$, and $\mathcal{N}(S_i)$ contains all one-hop neighbors of $S_i$. 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 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 $k = \tan\gamma$ is the slope, with $\gamma$ 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 $E(a, b, x_0, y_0, \nu) = 0$ [14]. Here $2a$ and $2b$ are the lengths of the major and minor axes of the ellipse, with $a, b$ drawn randomly from $[4,16]$. $(x_0, y_0)$ is the center of the ellipse, where $x_0$ and $y_0$ are randomly chosen from $[a, 64 - a]$. $\nu$, the angle between the major axis of the ellipse and the $x$-axis, is a random number between $0$ and $\pi$.

Note that both algorithms 1 and 2 need thresholds $\theta_1$ and $\theta_2$ to compute $C_1$ and $C_2$. Determination of these values is discussed in the next subsection.

### 3.6.2 Determination of Thresholds

**Simple thresholds determination**

One way [22] of obtaining the threshold values is to estimate them according to $p$, the probability that a sensor becomes outlying. For example, from the theory of Normal distribution, the ideal relationship between $\theta_1$ and $p$, may be seen partially in Table 3.1. Note 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. In the experiments, we first evaluate ALGORITHM 1 and ALGORITHM 2 based on the simple threshold settings as shown in Table 3.1. If $p = 0$, we simple set $\theta = 2$ in order to detect the event boundary.

<table>
<thead>
<tr>
<th>$p$</th>
<th>5%</th>
<th>10%</th>
<th>15%</th>
<th>20%</th>
<th>25%</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta_1$</td>
<td>1.96</td>
<td>1.65</td>
<td>1.44</td>
<td>1.28</td>
<td>1.15</td>
</tr>
</tbody>
</table>

Table 3.1: Relationship between $\theta_1$ and $p$ when $y$'s are iid from $N(0,1)$. 

35
Thresholds determination through the ROC curve analysis

The relationship between $\theta_1$ and $p$ in Table 3.1 is under the strict assumptions that the network has the information of $p$ in advance. However, due to practical violations of the assumptions, the actual relationship might deviate significantly from the one listed in the above 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 $\mathcal{N}$ and $\mathcal{NN}$ to compute $d$ value. This means that once we have an estimate for $\theta_1$, the same estimate will also be used for $\theta_2$. On the other hand, we propose to determine $\theta_1$ using an adaptive threshold determination scheme through the ROC (receiver operating characteristic) curve analysis [26].

The following is the basic 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 [7, 20]. Each time, the false alarm rates and detection accuracies based on one query are used to construct a 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 $\theta$. 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 $\theta$ is obtained from the knee point on the curve where the detection accuracy increases limitedly, whereas the false alarm rate exhibits a large boost. Base station will then broadcast the settings of the thresholds to the whole network.

In our simulation studies, we only consider the following six threshold settings: 3, 1.96, 1.65, 1.44, 1.28, 1.15, as shown in Table 3.1. For each query, the training data set contains
196 nodes, i.e., about 5% of the entire network. In the light of the ratio of the increase in detection accuracy to the increase in false alarm rate, one of these six settings will be selected as the optimal setting that corresponds to the knee point. Details are provided as follows.

Fig. 3.9 shows some ROC curves under network density of 30 and different values of \( p \), where the six points on each curve correspond to \( \theta = 3, 1.96, 1.65, 1.44, 1.28, 1.15 \) in the clockwise order. The detection accuracies and false alarm rates associated with the points on the curves are listed in Table 3.2, where for a given cell the first number represents the false alarm rate and the second number the detection accuracy. From 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 tradeoff, 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. 3.9. 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 can not be seen from either Fig. 3.9 or Table 3.2.) Nevertheless, the detection accuracy converts to increase slowly while 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 3.3 for various densities and values of \( p \). In the table, the five components in a column cell for a given
combination of the density and $p$ corresponds to $\theta_1 = 3.00, 1.96, 1.65, 1.44,$ and $1.28,$ respectively. Clearly, the angle derived from a line segment depends on the ratio of the increase in detection accuracy to the increase in false alarm rate. Table 3.3 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 $1.5$ (i.e., $86^\circ$), 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 density $= 30$ and $p = 0.15$. It follows from Table 3.3 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 3.3, is provided in Table 3.4. We see that the optimal threshold setting is smaller when there are more outlying sensors in the network, i.e., $p$ is larger.
<table>
<thead>
<tr>
<th>$\theta$</th>
<th>$p = 0.05$</th>
<th>$p = 0.10$</th>
<th>$p = 0.15$</th>
<th>$p = 0.20$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td></td>
<td>0.4444</td>
<td>0.1875</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>1.96</td>
<td>0.0053</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td></td>
<td>1.0000</td>
<td>0.6250</td>
<td>0.5000</td>
<td>0.2857</td>
</tr>
<tr>
<td>1.65</td>
<td>0.0107</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td></td>
<td>1.0000</td>
<td>0.8750</td>
<td>0.7500</td>
<td>0.5000</td>
</tr>
<tr>
<td>1.44</td>
<td>0.0214</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td></td>
<td>1.0000</td>
<td>0.8750</td>
<td>0.8571</td>
<td>0.5476</td>
</tr>
<tr>
<td>1.28</td>
<td>0.0374</td>
<td>0.0111</td>
<td>0.0060</td>
<td>0.0130</td>
</tr>
<tr>
<td></td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.8929</td>
<td>0.7381</td>
</tr>
<tr>
<td>1.15</td>
<td>0.0642</td>
<td>0.0222</td>
<td>0.0119</td>
<td>0.0455</td>
</tr>
<tr>
<td></td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9286</td>
<td>0.8571</td>
</tr>
</tbody>
</table>

Table 3.2: False alarm rates and detection accuracies associated with the points indicated on the ROC curves in Fig. 3.9.

<table>
<thead>
<tr>
<th>$p = 0.05$</th>
<th>$p = 0.10$</th>
<th>$p = 0.15$</th>
<th>$p = 0.20$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5612</td>
<td>1.5708</td>
<td>1.5708</td>
<td>1.5708</td>
</tr>
<tr>
<td>0</td>
<td>1.5708</td>
<td>1.5708</td>
<td>1.5708</td>
</tr>
<tr>
<td>0</td>
<td>1.5708</td>
<td>1.5708</td>
<td>1.5708</td>
</tr>
<tr>
<td>0</td>
<td>1.4821</td>
<td>1.4056</td>
<td>1.5027</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1.4056</td>
<td>1.3045</td>
</tr>
</tbody>
</table>

Table 3.3: Angles of line segments of ROC curves in Fig. 3.9.

<table>
<thead>
<tr>
<th>$\theta$</th>
<th>$p = 0.05$</th>
<th>$p = 0.10$</th>
<th>$p = 0.15$</th>
<th>$p = 0.20$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.96</td>
<td>1.44</td>
<td>1.44</td>
<td>1.28</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.4: Optimal threshold values derived from Table 3.3.

3.6.3 Simulation Results

In this subsection, we report our simulation results, each representing an averaged summary over 100 runs. More specifically, for each run of computation, the sensor field with or without an event region is generated using the procedure in subsection 3.6.1, the optimal threshold value is determined using the method described in subsection 3.6.2, and then Algorithm 1 or Algorithm 2 are applied to obtain the values of the performance metrics. The averaged values of the performance metrics over 100 runs are reported as
our simulation findings. The performance metrics include the detection accuracy and false alarm rate for outlying sensor detection, as defined by equation (3.4) in Subsection 3.5.1 for the evaluation of $C_1$, and the degree of fitting and false detection rate for event boundary detection, as defined by equations (3.5) and (3.7) in Subsection 3.5.3 for the evaluation of $C_3$. Recall that two sets, $C_1$ and $C_3$, contain the detected outlying sensors and boundary sensors respectively.

We note that based on the detection accuracy and 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 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 false detection rate could indicate that the event boundary has been successfully located. The related issue will be addressed in our future work.

We first evaluate our algorithm for outlying sensor detection based on different methods for threshold settings. Figs. 3.10 and 3.11 plot the detection accuracy versus false alarm rate $vs. \ p$, under different network densities. It is observed the two kinds of threshold settings generally achieve similar results for outlier detection. Adaptive method has larger detection accuracy and larger false alarm rate when $p = 10\%$ and density $\geq 20$ since a smaller threshold (1.44 versus 1.65) is set based on ROC curve analysis. Fig. 3.10 shows a general trend that the detection accuracy decreases as $p$ increases. From Fig. 3.11, we observe that when density equals to 20, 30, 40 and 50, the false alarm rate decreases as $p$ increases. From Figs. 3.10 and 3.11, 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 $\geq 30$, the detection accuracy is above 93.7% and false alarm rate is around 0.1% using adaptive threshold determination. For density $= 20$, the detection
accuracy is around 90% and false alarm rate is less than 1%. 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.

For event boundary detection described in Section 3.4, we need to set \( m \) in transition from \( C_1 \) and \( C_2 \) to \( C_3 \). 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. We first present the performance comparison for different threshold settings when density = 30 in Figs. 3.12 and 3.13. The results for other densities are similar. It is shown that the two kinds of
threshold settings obtain similar results, in degree of fitting and false detection rate.

Next, Figs 3.14 and 3.15 plot the experimental results for different densities through adaptive threshold settings. The main observations from these plots are summarized in the following:

a) 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.

b) 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.
Figure 3.14: Performance of the event boundary detection algorithm under different values of sensor outlying probability $p$ and network density (part-1).
c) 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 $\mathcal{N}^\prime$ so that more information is utilized in the detection process.

From Figs 3.14, 3.15, we see that the degree of fitting is low for cases where density equals 10. Note that a low degree of fitting does not mean that the boundary can not be detected. Instead it means that more sensors close to the boundary escape the detection. For example, Fig. 3.2 (d) indicates a case where the degree of fitting is as low as 53% 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%.)
We have also conducted simulations when input data are binary decision predicates and obtained results close to those reported in Figs 3.14, 3.15. This demonstrates that our algorithms are applicable to both 0/1 decision predicates and numeric sensor readings.
Chapter 4: Event Boundary Detection in Sensor Networks based on Gaussian mixture model

4.1 Introduction

In this section, we propose novel, more accurate and robust, statistic based algorithms for detecting event Boundary (or frontline) sensors [14, 22, 37, 40, 41, 63, 76]. Previous median based approaches [22, 76] work with scalar sensor inputs and thus only handles single channel information such as temperatures, or humidities over a geographical region. Previous works [14, 41] take 0/1 binary predicates only. Therefore it remains to be very problematic on how to process multivariate data resources under the condition of multi-modality sensor readings for event reporting [22, 76]. As a contrast, our new statistical model [1,2,28,67] based sensor information processing algorithms proposed in this chapter are capable of fusing multivariate real-valued sensor inputs to detect boundaries of events in a principled manner. Our new methods also naturally work for detecting boundaries of multiple event intersections, without constraining simplistic, individual event shapes. Our median based methods are designed to detect only simplified ellipse curves or straight lines as event boundaries/frontlines for the validity of their heuristic based random bi-section or tri-section Median schemes [22, 76].

Our first algorithm for event frontline detection is derived from Kmeans clustering with covariance deduction. The basic idea is based on the observation that a Boundary sensor is considered as residing within a local sensor neighborhood with a multimodal distribution of (univariate or multivariate) reading inputs, while each Non-Boundary sensor is surrounded with a neighborhood of unimodal sensor readings. Through Kmeans model fitting, the sensor nodes in a neighborhood set are clustered into different groups. Then, the boundary sensors can be classified with covariance deduction. And an outlying sensor can be detected by Mahanabolis distance. Our second algorithm for event boundary detection
is derived from statistical Gaussian mixture models with explicit model selection schemes [1, 2, 28, 67]. The distribution of sensor readings within each sensor’s spatial neighborhood is mathematically formulated using finite Gaussian Mixture Model [53]. The model selection techniques [1, 2, 28, 67] have been actively researched and developed in the past three decades. They can effectively identify the correct number of modes $\Gamma$ for finite mixture models, of which Gaussian mixture model is the most popular. Thus Boundary and Non-Boundary sensors can be consequently distinguished from its neighboring sensor data distributions of $\Gamma > 1$ or $\Gamma = 1$. Outlying sensor readings can be identified and treated naturally with extremely small density values due to the probabilistic nature of mixture models.

Our model based algorithms can be implemented within each purely localized neighborhood and thus scale well to large-range sensor networks. The computation overhead is moderate and comparable with Median based approaches [22, 76], but we achieve statistically much more comprehensive modeling capacity of the distribution of multivariate neighboring sensor readings. Extensive simulation results demonstrate that our proposed algorithms can accurately detect the event boundary sensors and also identify outlying sensors with high robustness regarding to various noise levels and different experimental settings. Though designed for sensor networks, our algorithms can be applied to general regional data analysis in spatial data mining [57] and network traffic mining [30, 51].

In summary, we propose two novel statistical algorithms for classifying Boundary and Non-Boundary sensors. Our first algorithm is based on Kmeans clustering with covariance deduction. The second classification method is based on the finite Gaussian mixture model [28, 53] and model selection techniques [1, 2, 28, 67].
4.2 Previous Work

4.2.1 Median based Event Boundary Sensor Detection [22, 76]

We first review our previous work on Event boundary sensor detection using Median computation [22, 76]. For representation clarity, we also follow the mathematical notation used in chapter 3. Median is a computation efficient robust estimator to extract the "real" sensing value from a spatially local set of sensors. It has the expected linear complexity against the size of the neighborhood set (ie. the number of sensors/readings).

The Median based algorithm contains the following four steps. 1), Each sensor $S_i$ computes the difference ($d_i = x_i - med_i$) between its own scalar reading $x_i$ and the median $med_i = Median(\{x_i\})$ of all sensor readings in its defined spatial neighborhood.

$$\mathcal{N}(S_i) = \{S_n\}$$

subject to : $\text{dist}(S_i, S_n) \leq D$  

(4.1)

, where $\text{dist}(S_i, S_k)$ measures either the Euclidean distance or one-hop, multi-hop distance between sensor $S_i$ and $S_k$; $D$ is the distance constraint to define the sensor neighborhood $\mathcal{N}(S_i)$.

2), In the sensor map, each node converts its real-valued difference $d_i, i = 1, 2, ..., m$ into Mahanabolis distance:

$$y_i = \frac{d_i - \hat{\mu}}{\hat{\sigma}}, i = 1, 2, ..., m$$  

(4.2)

as normalized by the estimated mean $\hat{\mu}$ and standard deviation $\hat{\sigma}$ from the global sensor set $\mathcal{N}^*$ including all $m$ sensors. The neighborhood relationship of $\mathcal{N}$ and $\mathcal{N}^*$ is illustrated in figure 4.1.

3), From the normalized difference $y_i$ of each sensor $S_i$, any outlying sensor can be
effectively detected by having distances $|y_i|$ above a certain threshold $\theta_1$.\footnote{In practice, we apply an adaptive threshold determination scheme to set the threshold, based on the ROC (receiver operating characteristic) curve analysis.} For an outlying sensor $S_i$, \begin{equation}
|y_i| > \theta_1 \tag{4.3}\end{equation}
. In [22, 76], outlying sensor detection is only designed for sensors living inside one single event.

4), For any sensor $S'_i$ lying on the event boundary or front-line, the set of its neighboring sensor readings contains two types of values satisfying two different (Normal) distributions of two event regions. Two neighborhood subsets $\mathcal{N}_j(S_i), j = 1, 2$ or $j = 1, 2, 3$ are generated using random bisection or trisection schemes, centering at each sensor $S_i$. Random bi-section randomly splits $\mathcal{N}$ as two half sub-regions, while trisection scheme divides $\mathcal{N}$ into three two-third area sub-regions. For details, refer to [22, 76]. Employing Median operator to compute $med_i$ inside each partition $\mathcal{N}_j(S_i)$ a new difference distance

\[ d_i = (x_i - med_i) \tag{4.4}\]

subject: $|x_i - med_i| = \max_j(|x_i - med_i|) \tag{4.5}$

is applied to compute normalized version $y_i$ using previous estimated $\hat{\mu}, \hat{\sigma}$ within any single event region. If

\[ |y_i| > \theta_2 \tag{4.6}\]

, where $\theta_2$ is another predefined threshold, $S_i$ is considered as event Boundary sensor, and vice versa.

If assuming that $x_i \in \mathcal{G}(\mu, \sigma^2)$ and $med_i$ is always chosen from samples of $\mathcal{G}(\mu, \sigma^2)$ (ie., $med_i = \text{median}(\mathcal{G}(\mu, \sigma^2))$), we can theoretically treat $d_i = x_i - med_i$ as a new random variable which satisfies another Normal distribution $\mathcal{G}(\hat{\mu}, \hat{\sigma}^2)$ where $\hat{\mu} = 0$ and $\hat{\sigma}^2 = 2\sigma^2$. The selection of thresholds $\theta_1$ and $\theta_2$ for the above constraints $|y_i| > \theta_1$, $|y_i| > \theta_2$ is also
based this Normality treatment, where $y_i$ is considered as the variant of $d_i$ from normalized Mahanabolis distance. Notice that the outlying sensor detection algorithm in [22, 76] only handle outlying failures inside a single sensing event region (as in step 3). There is no explicit method or analysis on dealing with outlying sensors while close to the event boundary or front-line as well (ie., outlying sensors). Bisecion or trisection strategies in [22, 76] work similarly. For representation simplicity, we will only discuss bisection scheme in the following sections.

4.2.2 Analysis

The algorithms described in [22, 76] have been empirically tested with satisfying experimental results using simulated data under relatively simple conditions, as compared later in this paper. They also attract much attention due to its simplicity and reasonably good performance. In this section, we further analyze its several intrinsic limits which motivates us to propose new algorithms rooted from finite Gaussian mixture model under a more statistical perspective.

Assume that $S_i$ is close the boundary of two events as Boundary sensor, its neighborhood $\mathcal{N}(S_i)$ thus includes the set of sensor readings $\mathcal{X}_i = \{x_k\}$, $S_k \in \mathcal{N}(S_i)$ from two Normal distributions $\mathcal{G}(\mu_1, \sigma_1^2)$ and $\mathcal{G}(\mu_2, \sigma_2^2)$. Then $med_i = median(\{x_k\})$, $S_k \in \mathcal{N}(S_i)$ is equivalent to $median(\mathcal{G}(\mu_1, \sigma_1^2), \mathcal{G}(\mu_2, \sigma_2^2))$ which can be very biased from either $median(\mathcal{G}(\mu_1, \sigma_1^2))$ or $median(\mathcal{G}(\mu_2, \sigma_2^2))$. Denote $\mathcal{G}_1$ as $\mathcal{G}(\mu_1, \sigma_1^2)$ and $\mathcal{G}_2$ as $\mathcal{G}(\mu_2, \sigma_2^2)$ For example, without loss of generality, we assume that $\mu_2 > \mu_1$. The median from two distributions has high probability as

$$median(\mathcal{G}_1) < median(\mathcal{G}_1 \cup \mathcal{G}_2) < median(\mathcal{G}_2)$$ (4.7)
The general assumption of [22, 76]

\[
\text{median}(G_1 \cup G_2) = \text{median}(G_j), \ j = 1, \text{ or } 2 \tag{4.8}
\]

is only approximately satisfied when the numbers of samples from \( G(\mu_1, \sigma_1^2), G(\mu_2, \sigma_2^2) \) are very unbalanced (which is unusual for \( \mathcal{N}(S_i) \) and \( S_i \) is close the boundary) or \( \max(\sigma_1, \sigma_2) \ll \min(\mu_1, \mu_2) \) (which is the case for the experiments in [22,76] as \( \mu_1 = 10, \mu_2 = 30 \) and \( \sigma_1 = \sigma_2 = 1 \)).

Under the assumption of equation 4.8 valid, we analyze the statistical conditions for both non-outlying event Boundary sensor and outlying Boundary sensor as follows. For a non-outlying Boundary sensor \( S_i' \), its reading \( x_i' \) follows either \( G_1 \) or \( G_2 \). Random Bi-section schemes [22, 76] are expected to find both \( \text{median}(G_1), \text{median}(G_2) \) as \( \text{med}_j, \ j = 1, 2 \) with the uniformity of sensor deployment assumption. Therefore equations 4.4, 4.5 become

\[
d'_i = (x_i' - \text{median}(G_j)) \tag{4.9}
\]

subject: \( |x_i' - \text{median}(G_j)| = \max_j(|x_i' - \text{median}(G_j)|) \tag{4.10}
\]

where \( j = 1, 2 \). More intuitively, this is also equivalent to

\[
d'_i = (x_i' - \text{median}(G_1)), \text{ if } x_i' \in G_2 \tag{4.11}
\]

or,

\[
= (x_i' - \text{median}(G_2)), \text{ if } x_i' \in G_1 \tag{4.12}
\]

Similarly, the effectiveness of random Bisection schemes and equations 4.10 strongly depends the validity of the general assumption as equation 4.8 within each partitioned subset \( \mathcal{N}_j(S_i), \ j = 1, 2 \), replacing the original set \( \mathcal{N}(S_i) \) from discussion above. For an outlying Boundary sensor \( S_i \) which is not addressed in [22,76], only equations 4.9, 4.10 (not 4.11, 4.12) hold. If its outlying reading value \( x_i \) is significantly different from both \( \text{median}(G_1) \) and \( \text{median}(G_2) \), \( S_i \) is likely to be detected.
figure 4.1: Relationship of a local sensor neighborhood \( N(S_i) \) and the global neighborhood \( N^* \). \( N(S_i) \) is the local neighborhood of sensor \( S_i \). If \( S_i \) is lying at the event boundary, there are two Normal distributions of sensor readings existing in \( N(S_i) \). Otherwise, one single normal distribution can well present all sensor values in \( N(S_i) \). The global neighborhood \( N^* \) is defined to be a large enough area within one single event region which includes sufficient number of \( N(S_i) \) for their \( \hat{\mu} \) and \( \hat{\sigma} \) statistics estimation of one single Normal distribution. Blue sensors are within \( N^* \) to be considered as one event, red sensors are lying outside of \( N^* \) and from another sensing event. \( N(S_i) \), containing both red and blue sensors, is treated as event boundary sensor.

It is well known that the sum \( x_1 + x_2 \) of two independent Gaussian random variables \( x_1 \) and \( x_2 \) is also Gaussian. Denote that the probability density functions of these two Gaussian distributions are \( G(\mu_1, \sigma_1^2) \), \( G(\mu_2, \sigma_2^2) \), then the sum variable \( x_1 + x_2 \) satisfies \( G(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2) \). [22, 76] assume that the Median estimator functions as a robust estimator of a set of sensor readings \( X_i = \{x_k\} \) within each spatial neighborhood \( N(S_i) = \{S_k\} \). For instance, if \( N(S_i) \) is a Non-Boundary sensor where all sensor values \( \{x_k\}, S_k \in N(S_i) \) are samples from \( G(\mu_1, \sigma_1^2) \), \( \text{med}_i \) or \( \text{Median}(X_i) \) is also considered a random variable \( \in G(\mu_1, \sigma_1^2) \). Therefore the random variable \( d_i = x_i - \text{med}_i \) satisfies the Gaussian as \( G(0, 2\sigma_1^2) \). In the other case that \( \{x_k\} \) contains samples from two Gaussian functions \( G(\mu_1, \sigma_1^2), G(\mu_2, \sigma_2^2) \) for an Boundary sensor \( S_i' \), algorithm in [22, 76] will find the difference variable \( d_i' \) as \( G(\mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2) \) if its reading \( x_i' \in G(\mu_1, \sigma_1^2) \), or \( G(\mu_2 - \mu_1, \sigma_1^2 + \sigma_2^2) \) if \( x_i' \in G(\mu_2, \sigma_2^2) \).
Finally the effectiveness of algorithms in [22,76] is directly based on whether the difference variables $d_i$ and $d'_i$ can be statistically distinguished. Using the Normality property, we obtain the effectiveness constraint of

$$|\mu_1 - \mu_2| > \theta_2(\sqrt{2}\sigma_1 + \sqrt{\sigma_1^2 + \sigma_2^2})$$ (4.13)

or

$$|\mu_1 - \mu_2| > \theta_2(2\sqrt{2}\sigma)$$ (4.14)

when $\sigma_1 = \sigma_2 = \sigma$ as [22, 76]. For an intuitive illustration, $|\mu_1 - \mu_2|$ needs to be larger than $8.49\sigma$ when $\theta_2 = 3$ for a good separability. This is a somewhat strict constraint which might degrade the practical performance of Median based algorithms [22,76] in noisy real environment. For the derivation of equations 4.13 and 4.14, refer to Appendix A.

**Normality of $N^*$**: The validity of the decision criteria for classifying outlying or event Boundary sensor (ie., equations 4.3, 4.2, 4.6) depends the Normality assumption of the underlying sensor value distribution from $N^*$. It also means that the estimated mean $\hat{\mu}$ and standard deviation $\hat{\sigma}$ out of $N^*$ is strictly constrained from one single Normal distribution of $N^*$. Without loss of generality, we denote this distribution as $G(\mu_1, \sigma_1^2)$. Thus the global neighborhood selection of $N^*$ can only cover a single sensor event area, but not the possible event boundary or front-line region where the difference variable

$$y_i \in G(\mu_1 - \mu_2, \sigma^2)$$

, or

$$y_i \in G(\mu_2 - \mu_1, \sigma^2)$$

$G(\mu_2, \sigma_2^2)$ is from any another sensing event region which adjacent to $N^*$ and $\sigma^2 = \sigma_1^2 + \sigma_2^2$. This can result in an chicken-and-egg problem. Detecting outlying or event boundary sensor requires of estimated $\hat{\mu}$ and $\hat{\sigma}$ from $N^*$, while defining a single event global neighborhood $N^*$ needs outlying or event boundary sensor to be detected and
removed as a prerequisite. Additionally, many parameters need to be set or tuned in [22,76], such as neighborhood settings as \( \mathcal{N}^* \), \( \mathcal{N} \), \( \mathcal{NN} \) and decision thresholds \( \theta_1 \), \( \theta_2 \). The settings of free parameters for optimal performance requires efforts and may still hazard the algorithms’ generality and applicability under new situations.

**Median of \( \mathcal{N}(S_i) \):** Median is a useful, but incomplete and limited description of any implying sensor value distribution from \( \mathcal{N}(S_i) \). It causes that the Median based event front-line detection algorithm in [22,76], theoretically only works validly for such neighborhoods \( \mathcal{N}(S_i) \) containing nearly half amount of sensors from one event region, and half from another. Median estimator/descriptor does not offer the modeling flexibility of defining an event neighborhood as including more than \( \Gamma \) portion of sensor values satisfying two or more distributions. In another word, there exist at least two distributions \( G(\mu_j, \sigma^2_j) \) in \( \mathcal{N}(S_i) \) with

\[
\text{prob}(X_i \in G(\mu_j, \sigma^2_j)) > \Gamma
\]  

(4.15)

, which motivates us to mixture modeling [28,53] in the next section. Furthermore, [22,76] assume some very simplistic geometry shapes as smooth line or ellipse of the event front-line separating any two events. In real environment, it can be possible that the shapes of event boundary are not just ideally lines or ellipses, or formed as a complex-shaped intersection of multiple event regions (ie., more than two). The performance of Median based methods will severely degenerate under above conditions. Finally, with more and more affordable sensor cost, the task of more robust and descriptive environment sensing can be achieved by deploying and fusing multiple channels of hybrid sensor readings, such as temperature, humidity, and others. This results that the new event Boundary detection algorithms needs to be capable of handling collections of multivariate, real-valued sensor reporting data. However, there is no principal way of modeling multivariate dependency and correlations using Median.
4.3 Motivation and Mixture Models

The mixture model has a wide variety of applications in practice, ranging from spatial data mining [57], large scale network dataflow monitoring and fault detection in complex network systems [30, 51], to effective face tracking [52], etc. Our major contribution is exploring the statistical modeling property of the finite mixture model [19, 28], especially the Gaussian mixture model, and adopting it into the scenario of distributive (sensor network) sensing data process and mining. We propose detecting the event frontline by model selection of mixture models.

4.3.1 Finite Mixture Models and Gaussian Mixture Models

Given a collection of data samples \( X = \{x_1, x_2, \ldots, x_m\} \) with each \( x_i \) representing a \( D \)-dimensional random (column) vector, assume that \( X \) follows a \( k \)-component (or mode) finite mixture distribution as

\[
p(x_i \mid \theta) = \sum_{j=1}^{k} \alpha_j p(x_i \mid \theta_j), \quad j = 1, 2, \ldots, k; \quad i = 1, 2, \ldots, m
\]

subject to: \( \sum_{j=1}^{k} \alpha_j = 1 \)

where \( \alpha_j \) is the mixing weight and \( \theta_j \) is the set of parameters of the \( j \)-th mixture component \( p(x_i \mid \theta_j) \). We denote \( \theta = \{\alpha_1, \theta_1, \alpha_2, \theta_2, \ldots, \alpha_k, \theta_k\} \) as the complete set of parameters defining a specific finite mixture model. The objective function of estimating \( \theta \) from \( X \) is to maximize the log-likelihood criterion

\[
logp(X \mid \theta) = \sum_{i=1}^{m} \log \left( \sum_{j=1}^{k} \alpha_j p(x_i \mid \theta_j) \right).
\]

(4.17)
Then the maximum likelihood estimator of $\theta$ is

$$\hat{\theta}_{ML} = \arg \max_{\theta} \{ \log p(X \mid \theta) \}. \quad (4.18)$$

It is well known [19, 28, 53, 62] that $\hat{\theta}_{ML}$ cannot be computed analytically from equation 4.18. Instead, expectation maximization (EM) algorithm [19, 62] is applied as its general solver for such a problem. EM is an iterative parameter optimization procedure of finding the maximum likelihood solution $\hat{\theta}_{ML}$.

The Gaussian mixture model (GMM) is the most important class of finite mixture densities [28]. GMM offers flexible and comprehensive unsupervised data modeling capacity and is capable of approximating any underlying density distribution in theory. GMM is formulated by using a Gaussian density $G(x_i \mid \mu_j, \Sigma_j)$ with its mean vector $\mu_j$ and covariance matrix $\Sigma_j$ to replace the general probability density function $p(x_i \mid \theta_j)$ in the finite mixture model

$$p(x_i \mid \theta) = \sum_{j=1}^{k} \alpha_j G(x_i \mid \mu_j, \Sigma_j), \quad (4.19)$$

where a $D$-dimensional multivariate Gaussian distribution is defined as

$$G(x \mid \mu, \Sigma) = \frac{1}{(2\pi)^{D/2} |\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (x - \mu)' \Sigma^{-1} (x - \mu) \right\}.$$  

4.3.2 Expectation Maximization Algorithm

The EM algorithm was originally presented for mixture model parameter estimation in Dempster’s paper [19] under the incomplete data condition. It was then extensively developed in the last three decades [28,53,62] and remains one of the most fundamental and important data modeling methods. EM algorithm is an iterative method on estimating the free parameters of finite mixture models, or more specific, the mean vectors $\mu_j$, covariance
matrices $\Sigma_j$ and prior weights $\alpha_j$ in GMMs.

The EM algorithm for GMM includes iterations of two steps as follows [28, 53, 62]. At each iteration $t$, both the E-step and M-step are executed.

**E-step:** Estimate the posterior probability $\gamma_{ij}^t$ that the j-th component generated $x_i$ using the estimates of the parameters from the last M-step estimation of the expected probability values of each $x_i$ conditioned on each mixture component, using the model $\hat{\theta}_{ML}^{t-1}$ obtained from the last maximization step.

$$
\gamma_{ij}^t = \frac{\alpha_{j}^{t-1} G(x_i | \mu_{j}^{t-1}, \Sigma_{j}^{t-1})}{\sum_{j=1}^{k} \alpha_{j}^{t-1} G(x_i | \mu_{j}^{t-1}, \Sigma_{j}^{t-1})}
$$

(4.20)

The probability normalization of $x_i$ with all mixture components

$$
\sum_{j=1}^{k} p(x_i | \hat{\theta}_{j}^{t-1}) = 1
$$

is satisfied to ensure the sum of contributions of each data sample $x_i$ for the estimated mixture model $\hat{\theta}$ is 1.

**M-step:** Maximization of $\log p(X | \hat{\theta}_{ML}^{t-1})$ to compute $\hat{\theta}_{ML}^{t}$ using $\gamma_{ij}^t$ from **E-step**. For each mixture component $j$, its mixing weight $\alpha_{j}^t$ is updated

$$
\alpha_{j}^t = \frac{\sum_{i=1}^{m} \gamma_{ij}^t}{\sum_{j=1}^{k} \sum_{i=1}^{m} \gamma_{ij}^t}
$$

(4.21)

where

$$
\sum_{j=1}^{k} \alpha_{j}^t = 1
$$
is guaranteed. Then its mean vector \( \mu_j^t \) is obtained

\[
\mu_j^t = \frac{\sum_{i=1}^{m} \gamma_{ij}^t x_i}{\sum_{i=1}^{m} \gamma_{ij}^t}
\]

(4.22)

and the covariance matrix \( \Sigma_j^t \) as

\[
\Sigma_j^t = \frac{\sum_{i=1}^{m} \gamma_{ij}^t (x_i - \mu_j^t)(x_i - \mu_j^t)'}{\sum_{i=1}^{m} \gamma_{ij}^t}.
\]

(4.23)

EM algorithm generates a sequence of model parameter estimates \( \hat{\theta}_{ML}^t, t = 1, 2, \ldots \), until convergence, by alternatively iterating E-step and M-step from an initialization of \( \theta^0 \). The most common initialization is to set \( \alpha_j = 1/k \), \( \Sigma_j = I(D) \) where \( I(D) \) is a \( D \)-dimensional identity matrix, and \( \mu_j = \text{random}(\{x_1, x_2, \cdots, x_m\}) \) for any mixture model component \( j \). Convergence of the EM algorithm is guaranteed since the objective function is monotonically increasing for each incremental iteration and the global optimum is bounded [53].

The model initialization and optimization convergence issues will be discussed later in this section. For the detailed derivation of equations 4.20, 4.21, 4.22 and 4.23, refer to [28, 53, 62].

4.3.3 Model Selection

The EM algorithm does not provide any information regarding selection of the number of mixtures from data. Clearly, such a selection is an important and unavoidable computational issue for GMM and EM because the maximum likelihood values obtained by EM, are increasing monotonically with larger numbers of mixture mode setting in GMM. The correctly estimated number of mixture components \( k \) also indicates important
statistical structures of the underlying distribution of the data, such as single resource \((k = 1)\) or multiple resource \((k > 1)\).

Akaike’s information criterion (AIC) [1, 2] and Bayesian information criterion (BIC) [67] are the two most popular model selection criteria based on penalty terms of model complexity. AIC and BIC intend to compute the model that best represents the data (i.e., data likelihood term), with a minimum number of free parameters to be estimated (i.e., model complexity term). Given \(\theta\) as a finite mixture model and \(X\) as data, AIC is defined

\[
AIC(\theta) = -2\log(p(X|\theta)) + 2K
\]  

(4.24)

where \(\log(p(X|\theta))\) is the log likelihood probability of \(X\) given \(\theta\) and \(K\) is the total number of free parameters in model \(\theta\) [1, 2]. The basic idea of model selection is to evaluate \(AIC(\theta)\) over a range of predefined mixture mode numbers \(k\), where the likelihood term \(-2\log(p(X|\theta))\) is computed after EM optimization of \(\theta\) and \(K\) is uniquely dependent on \(k\) given fixed covariance matrix setting. Then the \(k\) with the lowest \(AIC(\theta)\) score\(^2\) is selected as the mixture model fitting.

Nevertheless, AIC has the tendency of obtaining an estimated model with a negative bias towards the true data distribution when the data sample number \(m\) is in the same magnitude as \(K\) [36]. Here \(K\) is the total number of parameters to be estimated in GMM. Thus in this paper, we use the following BIC [67]

\[
BIC(\theta) = -2\log(p(X|\theta)) + K\log(m)
\]  

(4.25)

for GMM model selection. To use BIC for model selection, we simply choose the model that leads to the smallest BIC over the set of possible models. The only deference is that the model complexity term of BIC is better formulated as \(K\log(m)\) by considering both \(K\) and \(m\), compared with \(2K\) in AIC.

\(^2\)The estimated negative log-likelihood value \(-2\log(p(X|\theta))\) by EM, is monotonically decreasing as \(k\) increasing, but on the contrary the model complexity term \(2K\) increases simultaneously. It penalizes the larger model mode number \(k\) to avoid over-fitting.
There is another body of work on finite mixture models [28] who are capable of selecting the number of mixture modes $k$ during EM optimization. For details, refer to [28]. We leave this issue for future work.

4.3.4 Initialization and Convergence:

Since EM is an iterative optimization, it needs some initial settings for the GMM $\theta$ of $\{\alpha_j, \mu_j, \Sigma_j, j = 1, ..., k\}$ to start. The most common initialization is to assume a uniform prior of $\alpha_j = 1/k; \Sigma_j = I(D)$ where $I(D)$ is a $D$-dimensional identity matrix, and $\mu_j = random(\{x_i, i = 1, ..., m\})$ for any mixture model component $j$.

Convergence of EM algorithm is guaranteed because the objective function is monotonically increasing for each incremental iteration and the global optimum is bounded [53].

4.4 Algorithms

In this section, we describe our algorithms for event boundary sensor detection. The first algorithm classifies the boundary sensors by Kmeans clustering with covariance deduction. Our second algorithm is based on the general concept of modeling a collection of data samples using finite mixture models, especially GMMs. Specifically, we provide a solution for classifying event boundary sensors using EM based model fitting and model selection techniques. Our proposed algorithms have more profound statistical backgrounds than our previous counterparts [22, 76], and work well under more challenging and more realistic conditions of sensor deployment and sensor readings. They are also scalable, and purely distributed.

Given a sensor network $\{S_i\}$, we assume that sensors are moderately dense-deployed in the
spatial domain and the spatial distribution of sensing signals is smooth within each event
region. From a mathematical perspective, sensor readings provide a dense, but discrete
samplings of the underlying continuous distribution. Furthermore, $x_i$, the reading of $S_i$, is
considered as a $D$-dimensional random vector. To detect or classify the event boundary
sensors, the associated neighborhood sensor set $\mathcal{N}(S_i)$ of each sensor $S_i$ is constructed as
in equation 4.1.

4.4.1 Sensor Classification using Kmeans Clustering and Covariance Deduction

Our algorithms for boundary sensor classification are motivated on the observation that
a Boundary sensor is considered as residing within a local sensor neighborhood with a
multimodal distribution of (univariate or multivariate) reading inputs, while each Non-
Boundary sensor is surrounded with a neighborhood of unimodal sensor readings. Thus, the
basic solution is to cluster the sensor nodes in the neighborhood set into different groups.
Algorithm 3 consists of Kmeans model fitting, classifying event Boundary sensor by
covariance deduction and detecting Outlying sensor using Mahanabolis distance. Kmeans
is directly related to the $k$-center problem as described in Appendix B. It proceeds as
follows:

1. Given a data set of $m$ points $\{x_i, i = 1, 2, ..., m\}$ and a predefined cluster number $k$,
randomly select $k$ points $\in \{x_i\}$ as the initial $k$-centers $\{\mu_j, j = 1, 2, ..., k\}$.

2. Update Labels: For each data points $x_i$, assign its membership $w_{ij} \in \{0, 1\}$ by setting
$w_{ij} = 1$, if $dist(x_i, \mu_j) = \min_{j=1}^{k} dist(x_i, \mu_j)$; or $w_{ij} = 0$, otherwise.

3. Update $k$-centers: recalculate $\{\mu_j, j = 1, 2, ..., k\}$ by $\mu_j = (\sum_{i=1}^{m} w_{ij} x_i)/(\sum_{i=1}^{m} w_{ij})$.

4. Update $k$-weights: recalculate $w_j = \sum_{i=1}^{m} w_{ij}/m$ which satisfies $\sum_{j=1}^{k} kw_j = 1$.

\(^3\)In our approach, the pairwise correlations among multi-modality sensor readings, such as temperature, humidity and others, are
naturally formulated by the covariance matrices of finite/Gaussian mixture models.

61
Go to step 2, or exit until some convergence criterion meets.

The convergence criterion is defined as no change of memberships \( \{w_{ij}\} \) or centers \( \{\mu_j\} \). \textit{Kmeans} is generally considered as a hard-decision clustering problem, compared with \textit{EM} and \textit{GMM}. In this paper, algorithm 3 is developed as an intermediate solution between \textit{Medians} \cite{22, 76} and \textit{EM} based approaches, using explicit clustering to explore the set of neighboring sensor readings. \textit{Kmeans} can also handle multivariate sensor signals, but lack of theoretical model selection and probabilistic analysis.

**Algorithm 3** Event Boundary and Outlying Sensor Classification using Kmeans Clustering and Covariance Deduction

\[\begin{align*}
\text{inputs:} & \quad \text{A distribution of multi-variant sensor readings } x_i \text{ and } \{x_k\} \text{ of a sensor } S_i \text{ and its neighboring sensors } S_k \in \mathcal{N}(S_i) \\
\text{outputs:} & \quad \text{Labels } E(S_i), F(S_i) \text{ of sensor } S_i: \text{If } S_i \text{ is an Boundary sensor, set } E(S_i) = 1; \text{otherwise } E(S_i) = 0; \\
& \text{If } S_i \text{ is an Outlying sensor, set } F(S_i) = 1; \text{otherwise } F(S_i) = 0.
\end{align*}\]

1. Compute the mean vector and covariance matrix \( \mu, \Sigma \) from the set of its neighboring sensors’ readings \( \{x_k\} \) of \( \mathcal{N}(S_i) \).

2. Input the set of its neighboring sensors’ readings \( \{x_k\} \) of \( \mathcal{N}(S_i) \) into \textit{Kmeans} algorithm for model fitting under the component number is 2, then return the output \( \{\mu_j, \Sigma_j, w_j\}_{j=1,2} \) where \( \mu_j, \Sigma_j \) and \( w_j \) denote the mean, covariance and weight of \( j \)-th cluster component respectively.

3. \textit{Cluster Pruning:} Without loss of generality, we assume \( w_1 >= w_2 \). Then if \( w_2 < \omega_1 \) and \( ((\mu_1 - \mu_2)^T\Sigma_1^{-1}(\mu_1 - \mu_2))^\frac{1}{2} < \omega_2 \), set \( CL = 1 \) and \( E(S_i) = 0 \), and go to step 5; otherwise, set \( CL = 2 \) and go to step 4.

4. \textit{Classify Boundary Sensor by Covariance Deduction:} if \( (tr(\Sigma))^\frac{1}{2}/(w_1(tr(\Sigma_1))^\frac{1}{2} + w_2(tr(\Sigma_2))^\frac{1}{2}) > \omega_3 \), set \( E(S_i) = 1 \); otherwise set \( E(S_i) = 0 \).

5. \textit{Classify Outlying Sensor by Mahanabolis Distance:} if \( CL = 1 \) and then \( ((x_i - \mu)^T\Sigma^{-1}(x_i - \mu))^\frac{1}{2} > \omega_2 \), set \( F(S_i) = 1 \); if \( CL = 2 \) and then \( \min_{j=1}^2((x_i - \mu_j)^T\Sigma_j^{-1}(x_i - \mu_j))^\frac{1}{2} > \omega_2 \), set \( F(S_i) = 1 \); otherwise, \( F(S_i) = 0 \).

In application to sensor information processing, the main idea is that the weighted sum of within-cluster covariance will be reduced significantly from the global covariance without clustering, for a set of data readings from two or more underlying distributions (or distributions modes). Here the covariance magnitude is measured using the square-root of trace of covariance matrix. Due to the lack of essential model selection scheme of \( k \) for \textit{Kmeans} algorithm, we only compare \textit{Kmeans} with \( k = 2 \). We argue that this is
mostly sufficient to distinguish Boundary sensor with a multimodal (or multi-cluster) data neighborhood and Non-Boundary sensor inside a single-cluster reading set. Therefore we compute two hypothesized sets of mean vector and covariance matrix: 1), \(\{\mu, \Sigma\}\) using all \(\{x_k\}\) of \(N(S_i)\); 2), \(\{\mu_j, \Sigma_j, w_j\}_{j=1,2}\) output from \textit{Kmeans} algorithm with cluster number \(k = 2\).

Based on these two sets, we first perform \textit{Cluster Pruning} using two criteria:

\[
\begin{align*}
  \ w_2 & < \omega_1 \\
  ( \mu_1 - \mu_2)^T \Sigma_1^{-1} ( \mu_1 - \mu_2) & < \omega_2
\end{align*}
\]

where \(w_1 \geq w_2\). It means that the smaller cluster \(\{\mu_2, \Sigma_2, w_2\}\) will be removed if its overall weight \(w_2\) is significantly small (e.g., smaller than a preset threshold \(\omega_1\)) and its mean vector \(\mu_2\) is close enough to \(\mu_1\) (e.g., within \(\omega_2\) standard deviations of \(\{\mu_2, \Sigma_2, w_2\}\)). \(\omega_2\) is similar to Mahanabolis distance used in [22,76]. We can conclude that the true cluster number \(CL = 1\) and \(S_i\) is an Non-Boundary sensor and set \(E(S_i) = 0\). Otherwise \(CL = 2\), and the \textit{Covariance Deduction} constraint needs to be satisfied to further classify a sensor as an Boundary sensor for \(E(S_i) = 1\).

\[
(tr(\Sigma))^{\frac{1}{2}} / (w_1(tr(\Sigma_1))^{\frac{1}{2}} + w_2(tr(\Sigma_2))^{\frac{1}{2}}) > \omega_3
\]

where \(tr(\Sigma))^{\frac{1}{2}}\) is the square-root of the trace of the covariance matrix \(\Sigma\). It generally measures the magnitude of a multivariate covariance matrix. Therefore, if there are two valid clusters existing in the set \(\{x_k\}\) of \(N(S_i)\), the weighted sum of covariance magnitudes \((w_1(tr(\Sigma_1))^{\frac{1}{2}} + w_2(tr(\Sigma_2))^{\frac{1}{2}})\) after \textit{Kmeans} clustering is smaller than a fraction of the original \(tr(\Sigma))^{\frac{1}{2}} / \omega_3\). Finally an \textit{Outlying} sensor is detected of having \(> \omega_2\) standard deviations distance with \(\mu\) when \(CL = 1\), or with both \(\mu_1\) and \(\mu_2\) under \(CL = 2\), as an outlier to the underlying data distribution. All above steps are summarized in algorithm 3.
### 4.4.2 Sensor Classification based on Gaussian Mixture Model with Model Selection

In the following, we describe our sensor classification method leveraging on *GMMs* of soft, probabilistic clustering process as algorithm 4.

To check whether or not $S_i$ is a sensor lying on the boundary of an event, we input the data $\{x_n\}$ from readings of the sensors in $\mathcal{N}(S_i)$ and then build our best *GMM* based on $\{x_n\}$. Here the *EM* algorithm in Section 4.3.2 is applied for parameter estimation, and the *BIC* in Section 4.3.3 is used to select the final model. More details on this follow below.

We first set the upper bound of the mixture component number to be $K$. Then for each $J = 1, 2, \ldots, K$, the data set $\{x_n\}$ is fed into the *EM* algorithm for estimation of $\theta(J)$. Correspondingly, we obtain *BIC* scores $\{\text{BIC}(\theta(J))\}_{J=1,2,\ldots,K}$ in light of equation 4.25. Let $CL$ denote the number of mixture components of our final model (the best model). We select $CL = k$ where $\text{BIC}(\theta(k)) = \min_{J=1}^K \text{BIC}(\theta(J))$. Therefore our final is $\theta(\text{CL})$, or $\{\mu_j, \Sigma_j, \alpha_j\}_{j=1,\ldots,\text{CL}}$. Optionally, a sensitivity test can be performed by removing from $\theta(\text{CL})$ any under-presented mixture components with weights $\alpha_j < \zeta_1$. Then we obtain an updated model $\theta'(\text{CL})$ with $CL = CL - k_0$ where $k_0$ is the number of removed components from the original $\theta(\text{CL})$. Now it is straightforward to check the status of $S_i$. $S_i$ is treated as an event boundary sensor if and only if $CL > 2$. To classify if $S_i$ is an outlying sensor, the conditional probability for $x_i$ given model $\theta'(CL)$ is computed

$$p(x_i | \theta'(CL)) = \sum_{j=1}^{CL} w_j G(x_i | \mu_j, \Sigma_j)$$

(4.29)

then if $p(x_i | \theta'(CL)) < \zeta_2$, $S_i$ is classified as an outlying sensor and $\mathcal{F}(S_i) = 1$ is set; otherwise $\mathcal{F}(S_i) = 0$. Note that $\zeta_1$ is a threshold indicating an insignificant value of overall mixture component weight, and $\zeta_2$ is used as another threshold to measure outlying sensors which have significantly low probability density values given the final model $\theta'(CL)$. The above is summarized in algorithm 4. *BIC* [67] is chosen as the default technique. Other criterion as *AIC* [1, 2], or recursive *EM* algorithms with inherit model selection [28] can
Algorithm 4 Event Boundary and Outlying Sensor Classification based on GMM

**inputs:** Multivariate sensor readings \( x_n \) of sensors from the neighborhood \( N(S_i) \).

**outputs:** Labels \( E(S_i) \) of sensor \( S_i \). If \( S_i \) is an event boundary sensor, set \( E(S_i) = 1 \); otherwise \( E(S_i) = 0 \).

1. **Input** \( x_n \) to build GMM using EM algorithm (equations 4.19, 4.20, 4.21, 4.22 and 4.23) for parameter estimation and BIC (equation 4.25) for model selection.

2. **Mixture Component Number Determination:** Set the upper bound of the mixture component number to be \( K \). From the available GMMs \( \{\theta(J)\}_{J=1,2,...,K} = \{\{\mu_j, \Sigma_j, \alpha_j\}_{j=1,...,J}\}_{J=1,2,...,K} \) and their associated BIC scores \( \{BIC(\theta(J))\}_{J=1,2,...,K} \), we select \( CL = k \) where \( BIC(\theta(k)) = \min_{J=1}^{K} BIC(\theta(J)) \). Our final model is then \( \theta(CL) = \{\mu_j, \Sigma_j, \alpha_j\}_{j=1,...,CL} \).

3. **Sensitivity Testing (optional):** Denote by \( k_0 \) the number of mixture components with \( \alpha_j < \zeta_1 \). Removing these \( k_0 \) under-presented mixture components from \( \theta(CL) \) yields the new model \( \theta'(CL) \) with \( CL = CL - k_0 \).

4. **Classify Boundary Sensor:** If \( CL >= 2 \), \( S_i \) is an event boundary sensor and set \( E(S_i) = 1 \); otherwise \( E(S_i) = 0 \).

5. **Classify Outlying Sensor by density value:** Compute the conditional probability for \( x_i \) of sensor \( S_i \): \( p(x_i|\theta'(CL)) \) using equation 4.29. Then if \( p(x_i|\theta'(CL)) < \zeta_2 \), \( S_i \) is an Outlying sensor and set \( F(S_i) = 1 \); otherwise \( F(S_i) = 0 \).

Using BIC, we first set the upper bound of the mixture component number to be \( K \). Then under each \( J = 1,2,...,K \), the neighboring data set \( \{x_n\} \) of \( N(S_i) \) at sensor \( S_i \) is feeded into EM algorithm for parameter estimation of \( \theta(J) \). Finally we obtain BIC scores \( \{BIC(\theta(J))\}_{J=1,2,...,K} \) according to equation 4.25 for each model \( \theta(J) \); and select \( CL = k \) where \( BIC(\theta(k)) = \min_{J=1}^{K} BIC(\theta(J)) \). Denote that \( \theta(CL) \) is the final fitted model as \( \{\mu_j, \Sigma_j, w_j\}_{j=1,...,CL} \). Optionally, a sensitivity test can be performed as removing any under-presented mixture components with weights \( w_j < \zeta_1 \) from \( \theta(CL) \). Then we get updated model \( \theta'(CL) \) and \( CL = CL - k_0 \) where \( k_0 \) is the number of removed components of the original \( \theta(CL) \). For Boundary sensor classification using algorithm 4, we simply label \( S_i \) as a Boundary sensor and set \( E(S_i) = 1 \) if \( CL >= 2 \), or a Non-Boundary sensor with \( E(S_i) = 0 \) when \( CL = 1 \). To classify if \( S_i \) is an Outlying sensor, the conditional probability for \( x_i \) given model \( \theta'(CL) \) is computed using equation 4.29, then if \( p(x_i|\theta'(CL)) < \zeta_2 \), \( S_i \) is classified as an Outlying sensor and \( F(S_i) = 1 \) is set; otherwise \( F(S_i) = 0 \). Note that \( \zeta_1 \) is a threshold indicating an insignificant value of overall mixture
component weight, and $\zeta_2$ is used as another threshold to measure outlying sensors which have significantly low probability density values given the final model $\theta'(CL)$.

4.5 Experiments

In this section, we describe our experimental settings and report our evaluation results on the tasks of both Boundary/Non-Boundary sensor classification and outlying sensor identification. Numerical study and analysis of our newly proposed algorithms compared with previous works [22, 76] are also provided.

4.5.1 Experiment Setup

Our experiments are performed within a map consisting of $128 \times 128$ simulated sensors with each randomly deployed in one of the $128 \times 128$ grids, restricting one sensor per grid. To visualize a large sensor map with better clarity, we use an image to display the sensor map with multivariate sensing values. As shown in figure 4.2, we assume that sensors have multivariate sensing values from three different modalities, which form a three-component vector shown in RGB color. The extension of multivariate towards $>3$ dimensions is straightforward. In figure 4.2, (a) is a local sensor map of two sensing events shown in different colors on their spatial locations, while (b) is the corresponding global sensor map represented by an image. Note that (a) is a portion of (b) in the high-lighted blue rectangle area.

In our Matlab based experiments, we rigorously test the following signal models under different Normal distributions. The background sensing function is set as Gaussian noise $\mathcal{G}(\mu_0, \Sigma_0)$, where $\mu_0 = \text{randn}(3, 1) \times 8 + [25, 25, 25]'$ and $\Sigma_0$ is a symmetric, positive semi-definite matrix with diagonal elements as $\sigma$ and other elements as $\text{randn}(1, 1) \times \sigma \times 0.3$.
under $\Sigma_0(i, j) = \Sigma_0(j, i)$. Function $\text{randn}(m, n)$ returns a $m \times n$ matrix, of which each matrix element satisfies the standard Normal distribution $\mathcal{N}(0, 1)$. Diagonal elements of the covariance matrix represent self-correlation and off-diagonal elements indicate cross-correlation between pairwise multivariate variables. The use of multivariate Gaussian function (in our algorithms and experiments) provides a principled way of modeling and fusing multivariate sensor readings, which has never been investigated in previous works [14, 22, 41, 76]. In the experiments, we set $\sigma = 5$ by default. For synthesized sensor events, Normal distributions are also assumed. For a particular example of the $i$th event $\mathcal{G}(\mu_i, \Sigma_i)$, each variable in $\mu_i$ is randomly sampled from a 10-component vector $[25, 50, 75, 100, 125, 150, 175, 200, 225, 250]$ plus an additive Gaussian disturbance $\in \mathcal{N}(0, 8)$, to preserve the separatability among different sensing events including the background noise. $\Sigma_i$ is kept as the same of $\Sigma_0$ (ie., $\Sigma_i = \Sigma_0$) for simplicity.

There are three different shapes and five total configurations in our experiments. We randomly generate $q$ ellipses or rectangle bars at arbitrary orientations, or $q + 1$ radially divided zones in a star-graph in the simulated sensor map with $q$ event regions and
background region. In each event area $i$, sensor values are sampled from a specific multivariate \textit{Gaussian} distribution $\mathcal{G}(\mu_i, \Sigma_i)$ as discussed above. The lengths of ellipse axes are sampled uniformly from $[5, 20]$ independently. The center and orientation of each ellipse are randomly placed. For rectangle bar, we first produce a random line with length $> 20$ and then label all sensors as event sensors which are within a certain distance (randomly from $[5, 10]$) from the line. Due to the spatial randomness, an event $i$ may intersect with another event $j$. In this case, the simulated sensor readings of an overlapping area can be either selected from one of the two distributions according to an arbitrary order (i.e., one event distribution is overwritten by the other), or as the overlaid sum\(^4\). These two different ways of handling sensor readings in an overlapping region are denoted as $EL_{OW}$ or $EL_{OL}$ for the ellipse event shapes, and $LB_{OW}$ or $LB_{OL}$ for the event shape of rectangle bars, respectively. The star-graph originates at a random point in the center $32 \times 32$ region of the grid map, and its zones span some random angles from the range of $2\pi/(q + 1) \pm \pi/6$. Unless otherwise stated, we set $q = 3$ for experimental evaluations.

To evaluate the performance of algorithm 3 and 4, we use four classification accuracy metrics: event boundary rate (EB Rate), error rate (ER Rate), true positive rate (TP Rate), and false positive rate (FP Rate). EB Rate is the ratio of the number of \textit{Boundary} sensors to the number of \textit{all} sensors; ER Rate is the ratio of the number of \textit{incorrectly} classified sensors to the number of \textit{all} sensors; TP Rate is the ratio of the number of \textit{correctly} classified \textit{Boundary} sensors to the number of \textit{Boundary} sensors; and FP Rate is the ratio of the number of \textit{incorrectly} classified \textit{Boundary} sensors to the number of \textit{Non-Boundary} sensors.

\(^4\)This is equivalent to sampling from $\mathcal{G}(\mu_1, \Sigma_1) + \mathcal{G}(\mu_2, \Sigma_2)$, which is another multivariate \textit{Gaussian} function or component in the Gaussian mixture models. Our proposed algorithms are directly applicable in this situation.
4.5.2 Experimental Results

In this subsection, we first report the comparison results of our two modal based algorithms on boundary sensor classification without outlying sensor reading disturbance. The evaluation results of algorithm 4 are also present under the condition with outlying sensor presence.

**Examples without outlying sensors**

First, we evaluate the performance of *Boundary* sensor detection of algorithm 4 under the condition of no outlying sensors. For space reason we only report the results for \( EL_{OL} \), \( LB_{OL} \), and the star-graph event shapes in figure 4.3 (a.1-3), (b.1-3) and (c.1-3), respectively. The simulation results for \( EL_{OW} \) and \( LB_{OW} \) are similar. By default, the sensitivity parameter in algorithm 4 is set as \( \zeta_1 = 0.25 \). The upper bound of the component number is set as \( K = 5 \). The size of the neighborhood sensor set \( \mathcal{N}(S_i) \) used by all algorithms is 196. These parameters are kept as default in section 5.3 unless otherwise stated.

In figure 4.3, the (a-c.1) column shows the maps of simulated *multivariate Gaussian* data samples as sensor readings; the (a-c.2) column illustrates the ground truth boundary sensors\(^5\) shown in higher intensities; finally the (a-c.3) column is the corresponding detection results using our algorithm 4. *Boundary* sensors (shown in white) are considered as lying on the boundary of multiple (two or more including background noise) sensing events, while *Non-Boundary* sensors (shown in gray) existing inside the region of any single event. Black regions show areas where sensors do not have sufficient number of neighboring sensors to form the defined neighborhood set. For the examples in figure 4.3, the EB rates in the respective deployment maps are 7.46%, 15.27%, 11.80% and the ER rates are 1.69%, 2.04%, 0.66%. Note that figure 4.3 only shows the classification results in

---

\(^5\)\( \zeta_1 = 0.25 \) is used to mark out *True Boundary* sensors. A sensor \( \mathcal{N}(S_i) \) is labeled as a *True Boundary* sensor if its neighborhood sensor reading set \( \{x_n\} \) contains samples of at least two distributions and each has a ratio \( \geq 0.25 \).
one particular trial of sensor reading simulation. Refer table 4.1 for the performance based on the average of 100 runs.

**Comparison with Median based method in Chapter 3**

We compare algorithm 3 and 4 on Boundary sensor detection with the Median based method in Chapter 3 using bi-section scheme (which is claimed better than tri-section in [76]). The valid distributional assumption of Boundary sensor [22, 76] is defined as having exactly two mixture components with each weighting 0.5 ($w_j \approx 0.5$). Here we set $\zeta_1 = 0.45$ to label the ground truth for the Median based method due to statistical stability consideration. $w_j \approx 0.5$ is too strict to have sufficient numbers of Median classified Boundary sensors. Therefore the EB rates are noticeably less than the above case of our algorithm 4 where $\zeta_1 = 0.25$, but ER, TP and FP Rates are still informative performance metrics to be compared. The other parameter settings are the same as default.

Table 4.1 shows numerical error comparison of different algorithms with the average of 100 runs under five event shape configurations when $q = 3$. Both algorithm 3 (denoted as $KM_{CD}$) and 4 (denoted as $EM_{BIC}$) outperform the previous Median based methods (denoted as Median) [22, 76] in all five event configurations, with a statistically significant margin of higher TP Rates but lower ER and FP Rates. Algorithm 3 in section 4.4.1 performs the second as a non-probabilistic, statistical approximation to Gaussian mixture model used in algorithm 4.

**Robustness against outlying sensors**

To test the robustness of algorithm 4 against outlying sensors, we randomly choose a portion $\varpi$ (eg., 5%, 10%, ...) of sensors and add a noise offset as $\text{randn}(3, 1) \times 30$ to each of their original readings. The readings of other sensors and the parameter settings are unchanged. Table 4.2 shows the performance evaluation of our algorithm 4 under different outlying sensor ratios $\varpi$ in $EL_{OL}$. All results are the averages of 100 trials. Performance
Figure 4.3: Examples of event Boundary and Non-Boundary sensor classification using Algorithm 4 under different event configurations. Rows (a.1-3), (b.1-3), (c.1-3) are for $EL, OL, LB, OL$ and star graph event shapes respectively. The (a-c.1) column shows the maps of simulated multivariate Gaussian samples as sensor readings; (a-c.2) column describes the ground truth boundary sensors shown with higher intensities; (a-c.3) column is the according classification results using algorithm 4.
<table>
<thead>
<tr>
<th>Shape</th>
<th>Algorithm</th>
<th>EB Rate</th>
<th>ER Rate</th>
<th>TP Rate</th>
<th>FP Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL_OW</td>
<td>EM_BIC</td>
<td>0.0906</td>
<td>0.0019</td>
<td>0.9902</td>
<td>0.0012</td>
</tr>
<tr>
<td></td>
<td>KM_CD</td>
<td>0.0892</td>
<td>0.0201</td>
<td>0.9211</td>
<td>0.0176</td>
</tr>
<tr>
<td></td>
<td>Median</td>
<td>0.0179</td>
<td>0.034</td>
<td>0.6438</td>
<td>0.0211</td>
</tr>
<tr>
<td>LB_OW</td>
<td>EM_BIC</td>
<td>0.1712</td>
<td>0.0232</td>
<td>0.9632</td>
<td>0.0042</td>
</tr>
<tr>
<td></td>
<td>KM_CD</td>
<td>0.1853</td>
<td>0.0407</td>
<td>0.9133</td>
<td>0.0295</td>
</tr>
<tr>
<td></td>
<td>Median</td>
<td>0.0304</td>
<td>0.0692</td>
<td>0.6354</td>
<td>0.0436</td>
</tr>
<tr>
<td>EL_OL</td>
<td>EM_BIC</td>
<td>0.0844</td>
<td>0.0123</td>
<td>0.9722</td>
<td>0.0084</td>
</tr>
<tr>
<td></td>
<td>KM_CD</td>
<td>0.0858</td>
<td>0.0258</td>
<td>0.8989</td>
<td>0.0176</td>
</tr>
<tr>
<td></td>
<td>Median</td>
<td>0.0215</td>
<td>0.035</td>
<td>0.7651</td>
<td>0.0247</td>
</tr>
<tr>
<td>LB_OL</td>
<td>EM_BIC</td>
<td>0.1527</td>
<td>0.0204</td>
<td>0.9646</td>
<td>0.009</td>
</tr>
<tr>
<td></td>
<td>KM_CD</td>
<td>0.1662</td>
<td>0.0432</td>
<td>0.8678</td>
<td>0.0296</td>
</tr>
<tr>
<td></td>
<td>Median</td>
<td>0.0243</td>
<td>0.0698</td>
<td>0.6108</td>
<td>0.0461</td>
</tr>
<tr>
<td>Star</td>
<td>EM_BIC</td>
<td>0.118</td>
<td>0.0066</td>
<td>0.9713</td>
<td>0.0036</td>
</tr>
<tr>
<td></td>
<td>KM_CD</td>
<td>0.1123</td>
<td>0.0387</td>
<td>0.9346</td>
<td>0.0168</td>
</tr>
<tr>
<td></td>
<td>Median</td>
<td>0.02</td>
<td>0.0519</td>
<td>0.2462</td>
<td>0.0283</td>
</tr>
</tbody>
</table>

Table 4.1: Comparison of Median based method [22, 76], algorithm 3 and algorithm 4 under five Event Shapes.

of algorithm 4 drops gracefully with increasing $\omega$. The TP Rates of Boundary sensors remain at the same level, but more false alarms (FP Rates) appear. We also test the robustness of Median based algorithms for comparison purpose. It degrades about 2.13 times faster than our mixture model approach in FP Rate. Due to the space limitation, we omit numerical details. Results of other four cases are similar.

Importantly, the detection rates (eg., recalls) of outlying sensors by using algorithm 4 are also reported in table 4.2 (as OR Rate). This result shows slightly worse performance but comparable stability than boundary sensor detection as $\omega$ increases. Our mixture model based framework is capable to provide an unified approach for simultaneous boundary and outlying sensor detection.
Robustness Evaluation on the outlying sensor ratios $\varpi$ of Algorithm 4

<table>
<thead>
<tr>
<th>Metric</th>
<th>$\varpi = 0%$</th>
<th>$\varpi = 5%$</th>
<th>$\varpi = 10%$</th>
<th>$\varpi = 15%$</th>
<th>$\varpi = 20%$</th>
</tr>
</thead>
<tbody>
<tr>
<td>EB Rate</td>
<td>0.0844</td>
<td>0.0859</td>
<td>0.0864</td>
<td>0.0834</td>
<td>0.0858</td>
</tr>
<tr>
<td>ER Rate</td>
<td>0.0123</td>
<td>0.0148</td>
<td>0.0159</td>
<td>0.0193</td>
<td>0.0264</td>
</tr>
<tr>
<td>TP Rate</td>
<td>0.9722</td>
<td>0.9691</td>
<td>0.9519</td>
<td>0.9583</td>
<td>0.9562</td>
</tr>
<tr>
<td>FP Rate</td>
<td>0.0084</td>
<td>0.0108</td>
<td>0.0122</td>
<td>0.0179</td>
<td>0.0231</td>
</tr>
<tr>
<td>OR Rate</td>
<td>——</td>
<td>0.9481</td>
<td>0.9238</td>
<td>0.9029</td>
<td>0.9185</td>
</tr>
</tbody>
</table>

Table 4.2: Robustness evaluation on the outlying sensor ratios $\varpi$ of algorithm 4 in EL\_OL.

4.5.3 Additional Testing & Evaluation

In this section we evaluate the performance of algorithm 4 on different parameter settings, i.e., threshold $\zeta_1$, covariance scale $\sigma$, and neighborhood size $\phi$. We investigate all five configuration scenarios. In each experiment, we set the parameter to be evaluated with different values while keeping the other parameters as default. Due to space limit, we take $EL\_OL$ as the illustrating case of error analysis below. Performances in other four conditions are similar based on our experiments. The numerical results are the averages of 100 trials.

Performance Sensitivity and Flexibility Testing on $\zeta_1$

Table 4.3 shows the performance evaluation of algorithm 4 on different settings of parameter $\zeta_1$. As shown in the table, the Boundary and Non-Boundary sensor classification accuracy is insensitive to $\zeta_1$. When $\zeta_1$ increases, for example from 0.15, 0.25, 0.35 to 0.45, smaller EB Rates and thinner Boundaries are obtained, which is as expected. However at the meanwhile, the classification accuracy metrics of ER Rates, TP Rates and FP Rates remain at the same level as in algorithm 4 for $EL\_OL$, which shows the good stability over different $\zeta_1$s.

Performance Sensitivity on Covariance Scales

In our experiments, all Gaussian signal components (including background white noises)
Sensitivity and Flexibility Testing on $\zeta_1$ of Algorithm 4

<table>
<thead>
<tr>
<th>Metric</th>
<th>$\zeta_1 = 0.15$</th>
<th>$\zeta_1 = 0.25$</th>
<th>$\zeta_1 = 0.35$</th>
<th>$\zeta_1 = 0.45$</th>
</tr>
</thead>
<tbody>
<tr>
<td>EB Rate</td>
<td>0.1707</td>
<td>0.0844</td>
<td>0.0678</td>
<td>0.0262</td>
</tr>
<tr>
<td>ER Rate</td>
<td>0.0081</td>
<td>0.0123</td>
<td>0.0109</td>
<td>0.0082</td>
</tr>
<tr>
<td>TP Rate</td>
<td>0.9671</td>
<td>0.9722</td>
<td>0.9695</td>
<td>0.9794</td>
</tr>
<tr>
<td>FP Rate</td>
<td>0.0033</td>
<td>0.0084</td>
<td>0.0054</td>
<td>0.0046</td>
</tr>
</tbody>
</table>

Table 4.3: Performance evaluation on the different $\zeta_1$ settings of algorithm 4 in $EL, OL$.

| Performance Sensitivity Testing on $\sigma$ of Algorithm 4 |
|------------------|------------------|------------------|------------------|
| Metric           | $\sigma = 2$     | $\sigma = 5$     | $\sigma = 10$    | $\sigma = 15$    |
| EB Rate          | 0.0823           | 0.0844           | 0.0850           | 0.0838           |
| ER Rate          | 0.0120           | 0.0123           | 0.0127           | 0.0125           |
| TP Rate          | 0.9768           | 0.9722           | 0.9699           | 0.9703           |
| FP Rate          | 0.0077           | 0.0084           | 0.0101           | 0.0092           |

Table 4.4: Performance sensitivity testing on covariance scales $\sigma$ of algorithm 4 in $EL, OL$.

share the same covariance matrix $\Sigma_0$, which is defined in section 5.3.1 and $\sigma$ is a key factor controlling the noise levels. Thus we test the performance sensitivity of algorithm 4 with different $\sigma$ settings in table 4.4. Using $EL, OL$ as an example, we conclude that our algorithm 4 is very insensitive to $\sigma$ changes. No statistically significant performance downgrading is observed when $\sigma$ varies from 2 to 5, 10 and 15.

Performance Sensitivity on Neighborhood Size

We evaluate the performance sensitivity on sensor neighborhood size $\phi$ of algorithm 4 in $EL, OL$. From table 4.5, by varying $\phi$ from 36, 100, 196 to 400, the $EB$ Rates monotonically increase while the $ER$ and $FP$ Rates stay at the same level of $\phi = 36, 100, 196$, and raises slightly when $\phi = 400$. The differences of $TP$ are ignorable. This indicates that the impact of the changes of sensor neighborhood sizes $\phi$ on algorithm performance is stable within a moderate range of [36, 196]. When $\phi$ is extremely small, no sufficiently enough sensors are available for $EM$ based statistical $GMM$ algorithms.

Robustness on Exponentially Decaying Signals and Outlying Sensors

74
Performance Sensitivity Testing on \( \phi \) of Algorithm 4

<table>
<thead>
<tr>
<th>Metric</th>
<th>( \phi = 36 )</th>
<th>( \phi = 100 )</th>
<th>( \phi = 196 )</th>
<th>( \phi = 400 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>EB Rate</td>
<td>0.0630</td>
<td>0.0646</td>
<td>0.0844</td>
<td>0.1270</td>
</tr>
<tr>
<td>ER Rate</td>
<td>0.0126</td>
<td>0.0124</td>
<td>0.0123</td>
<td>0.0209</td>
</tr>
<tr>
<td>TP Rate</td>
<td>0.9706</td>
<td>0.9796</td>
<td>0.9722</td>
<td>0.9743</td>
</tr>
<tr>
<td>FP Rate</td>
<td>0.0086</td>
<td>0.0084</td>
<td>0.0084</td>
<td>0.0187</td>
</tr>
</tbody>
</table>

Table 4.5: Performance sensitivity testing on sensor neighborhood sizes \( \phi \) of algorithm 4 in EL.OL.

In EL.OL, to simulate the exponentially decaying signal factor, we product each multivariate sensor reading (a sample from \( G(\mu_i, \Sigma_i) \)) at location \((x, y)\) in event region \(i\), with a weight \( \exp(-d((x, y), (cx_i, cy_i))/\lambda) \). \( d((x, y), (cx_i, cy_i)) \) computes the Euclidean distance between \((x, y)\) and \((cx_i, cy_i)\) which is the center position of event \(i\), where parameter \( \lambda \) controls the decaying speed. Interestingly, our algorithm 4 shows its bi-modal performance phenomenon affected on \( \lambda \)'s changes. The maximal (decayed) weighting ratio within each 7-by-7 neighborhood set is approximately \( 1/\exp(-7/\lambda) \), which increases with decreasing \( \lambda \). 1) If \( \lambda > \lambda_1 \), our algorithm 4 can handle exponentially decaying signals as naturally distributional variations, without noticeable performance degrading. 2) Otherwise if \( \lambda \leq \lambda_1 \), the weight \( \exp(-d((x, y), (cx_i, cy_i))/\lambda) \) can introduce too substantial distortions on the original distribution of \( G(\mu_i, \Sigma_i) \), which causes our mixture model based algorithm to classify more (until all with smaller \( \lambda \)s) sensors inside event \(i\) as Boundary sensors. Empirically, \( \lambda_1 = 3.6 \times \phi \) where \( \phi \) is the local sensor neighborhood size and set as \( \phi = 7 \). This results the maximal decaying ratio as \( \approx 0.75 \) or above, which guarantees the satisfactory performance\(^6\) of algorithm 4. Illustrative examples of the above two cases are demonstrated in figure 4.4 when \( \lambda = 4.0 \times \phi \) in (a) or \( \lambda = 2.5 \times \phi \) in (b). In practice, we expect the decaying ratio to be much less than 0.75 within one local sensor neighborhood because the normal event regions are spatially smooth and with significantly larger scales than \( \phi \).

\(^6\)Under the rare case of \( \lambda > \lambda_1 \), algorithm 4 can be used as the pre-filtering, followed by an exponential function fitting process to detect regional, exponentially decaying signals. This process may require a larger scale of sensor readings for detectability, and thus is be more computationally expensive and not totally localized.
Figure 4.4: Illustrative examples of the bi-modal performance phenomenon of algorithm 4 when $\lambda = 4.0 \times \phi$ in (a) or $\lambda = 2.5 \times \phi$ in (b).
Chapter 5: Target Tracking in Sensor Networks based on Gaussian mixture model

5.1 Introduction

Target tracking has emerged as an interesting problem representing a very important class of sensor network applications [7, 32, 34, 43, 83]. In this chapter, we propose a robust (fault-tolerant) statistic algorithm for the task of tracking dynamically moving targets. Our design tackles the challenges faced by real time target tracking. We adopt an adaptive Gaussian mixture representation to handle the target mobility issue and employ the Mean-shift [16, 48] continuous optimization method for target localization. The nice features of the algorithm are summarized as follows:

1. Our novel statistical Gaussian mixture model based algorithm is capable of fusing multivariate real-valued sensor signals and model the sensing data distribution to capture the target presence.

2. Our approach is fault-tolerant and is robust against outlying sensor readings under complex background noise models because an adaptive model updating procedure is employed to dynamically update the target signature.

3. Given the estimated target mixture model, Mean-shift iterations are employed to localize the target’s new position at the next step.

Leveraging on the fast, reliable convergence and continuous optimization of mean-shift, and the probabilistic smoothness of the Gaussian mixture model, our tracking algorithm is capable of handling different types of target signal models under various noise disturbances. The issues of accuracy, smoothness, and robustness have significant practical importance, but have not been fully addressed in previous work [7, 22, 32, 34, 43, 83].
Extensive simulation results are also reported in this chapter, which demonstrate that our proposed algorithm can accurately track the target motion trajectory with a high robustness under various background noise patterns and levels. Though designed for sensor networks, we believe that our algorithm can be applied to general regional data analysis in spatial data mining [57] and network traffic mining [30, 51].

5.2 Algorithms

In this section, we describe our algorithm for dynamic target tracking under noisy sensing environment. Our algorithm is based on the general concept of modeling a collection of data samples using finite mixture models, especially GMMs. When a target enters into the monitored area, a group of sensors may detect the presence of the target. Many researches have explored collaborative information processing techniques [43, 83] to in-network process the readings of the group of sensors to achieve more accurate target detection and localization results. Our approach is in the same spirit, but we propose a dynamic, adaptively updated Gaussian mixture model to capture the data distribution from the group of sensors (which detect the target) over time. Our temporally adaptive version of GMM presents the target temporal appearance (in terms of sensing data distribution) and reflects their changes more effectively than the original “calibrated-then-fixed” GMM approach in [43,83]. To locate the target, the mean-shift algorithm [16,48] is adopted to find the group of sensors having the highest total probability density responses to the previous target distribution (parameterized by adaptive GMM). The mean-shift method continuously optimizes the location of the target in an iterative fashion, and provides fast and reliable convergence [16].

In summary, the target tracking problem is tackled with a two-step approach: using adaptive GMM to capture the distributional appearance of the target across both the spatial domain and over time; and then employing the mean-shift algorithm [16,48] to find the new target.
location.

5.2.1 Model Initialization for Target Appearance Distribution

Given a sensor network $G$, we assume that the set of sensors $\{S_i\}$ in $G$ are moderately densely-deployed in the spatial domain and the spatial distribution of sensing signals is smooth within the network region. From a mathematical perspective, sensor readings provide a dense, but discrete samplings of the underlying continuous signal distribution. Furthermore, $x_i$, the reading of $S_i$, is considered as a $D$-dimensional random variable or vector. In our approach, the pairwise correlations among multi-modality sensor readings such as temperature, humidity and others, are naturally formulated by the elements of covariance matrices of finite/Gaussian mixture models. The sensor nodes may become outlying\textsuperscript{1} due to failure hardware or harsh environment. Suppose that the outlying readings are uncorrelated in the network.

To save network energy, we assume some power management protocols [32] are employed to put some of the sensor nodes in “sleep” state when there are no targets in the network field. The sensing coverage is guaranteed that there is at least one sensor node in “active” state which can detect the target when a target presents in the monitored area. Then a group of sensor nodes will be waked up to collaboratively detect and locate the target. We call these sensor nodes event sensors. Thus a static target can be initially defined as the sensor readings of event sensors, and these readings form a specific statistical model.

We adopt the Gaussian mixture model to describe the distribution of the sensing signals capturing the existence of a target. These signals are called target signals and they jointly define the “appearance” of the target “seen” by the sensors. GMM is adopted for its modeling capacity and flexibility [28, 53]. Suppose the target signals can be modeled as the weighted sum of some Gaussian functions, ie., a Gaussian mixture model as

\textsuperscript{1}A sensors reading is outlying if it deviates significantly from other readings of neighboring sensors [4].
Eq. (4.19). Then the target can be parameterized by the set of parameters of the Gaussian mixture model using the EM algorithm. We denote this model as the target’s appearance model $\mathcal{A}$. To initialize the parameters of $\mathcal{A}$, we plug the readings of the event sensors in the EM algorithm and employ the BIC criterion for model selection, as discussed in subsection 4.3.2 and 4.3.3. We assume that there is a leader node or cluster head among the group of event sensors that implements the model initialization. All the event sensors transmit their measurements and locations to the leader node. The target’s initial location and effective spatial coverage is also known at the time frame 0 before tracking is employed.

5.2.2 Mean-shift Optimization for Target Localization

After we initialize the target appearance model $\mathcal{A}$, our next step is to track the target trajectory. We adopt the mean-shift optimization to discriminate the target signals against the general background noise and estimate the target position. Let $L(t - 1)$ denote the location of target event at time $(t - 1)$. $R(t - 1)$ is the set of event sensors that will be used to update model $\mathcal{A}(t - 1)$ at time $(t - 1)$. $R(t - 1)$ is geometrically reconstructed as a sensor neighborhood set around $L(t - 1)$. For simplicity, we assume that all sensors in $R(t - 1)$ reside in a disk centered at $L(t - 1)$, denoted by $D$. Or, we can intuitively consider $R(t - 1)$ as a “disk” centered at $L(t - 1)$. We further assume that the ship and size of $D$ remains unchanged though the target is keeping on moving. To track the moving target in the next time frame $t$, more sensor nodes around the predicted target location at the current time slot need to be waken up.

In our design, we activate all the sensor nodes located in a disk region that is 4 times larger than $D$ (i.e., expanding $D$ by two times in both $x$ and $y$ coordinates), but has the same center as $D$. Denote the set of active nodes for target tracking at time frame $t$ as $\mathcal{R}\mathcal{R}(t)$. Suppose that the leader of $\mathcal{R}(t - 1)$ can predict the target location for next time frame based on the historical trajectory of the target, and we can employ more sophisticated principles
(Kalman filter or Particle filter) to model the spatial dynamics of the trajectory. How to choose the optimal $RR(t)$ according to the target moving velocity and direction is left as future work. Figure 5.1 (a) shows a scenario to illustrate the set of event sensors $R(t - 1)$ and the set of active sensors $RR(t)$ awaken to monitor the target movement in successive time frames.

When the target moves to a new position at time frame $t$, some active nodes in $RR(t)$ can detect the target. Now the problem is how to define and localize the target’s new location $L(t)$. Below we describe the computational procedure of acquiring $L(t)$ based on the evaluation of the mixture models $A(t - 1)$ and the mean-shift algorithm [16]. In addition, to indicate the dependence on time, we use $x_i(t)$ to denote the reading of sensor $S_i$ at time $t$. Given $A(t - 1)$, we compute the value of the fitness by evaluating the probability density functions of the target appearance model $A(t - 1)$ for the reading $x_i(t)$ of each sensor $S_i$ in $RR(t)$. These values, denoted by $\{p(x_i(t) | A(t - 1))\}$ and appearing as a target-conditional response map (conditioned on $A(t - 1)$), can then be used to form an estimate of $L(t)$ using the mean-shift algorithm. Specifically, the mean-shift algorithm is an iterative process

$$
\hat{L}(t, (it + 1)) = \frac{\sum_{S_i \in \hat{R}(t, it)} p(x_i(t) | A(t - 1))L_i}{\sum_{S_i \in \hat{R}(t, it)} p(x_i(t) | A(t - 1))},
$$

(5.1)

where $L_i$ denotes the location of sensor $S_i \in \hat{R}(t, it)$. Here $\hat{R}(t, it)$ is the set of “event” sensors that contribute to the newly estimated target location at each iteration $it$. $\hat{R}(t, (it + 1))$ is then updated using $\hat{L}(t, (it + 1))$, in the same way as obtaining $R(t - 1)$ from $L(t - 1)$, or simply, by “moving” $\hat{R}(t, (it))$ to the new location of $\hat{L}(t, (it + 1))$. Note that $\hat{L}(t, (it + 1))$ is the estimated target location, which is the weighted “average” of the locations of the current set of event sensors in $S_i \in \hat{R}(t, it)$ given $p(x_i(t) | A(t - 1))$ from previous step. At the beginning of the iterative process, $\hat{R}(t, 1)$ is the same as $R(t - 1)$. For the subsequent iterations, $\hat{R}(t, (it + 1))$ and sensors $S_i \in \hat{R}(t, (it + 1))$ are updated using $\hat{L}(t, (it + 1))$ as follows. We simply choose the active sensor nodes reside in the
covering range of $\hat{L}(t, (it + 1))$ as event sensors. When convergent, the final $\hat{L}(t, (it + 1))$ is used to estimate the location $L(t)$. Refer Appendix C for detailed algorithm of kernel mean-shift target tracking. Empirically, the mean-shift converges within $3 \sim 5$ iterations in our experimental settings. Figure 5.1 illustrates the process of the mean-shift optimization for target tracking and localization. In contrast to our approach, [7, 43, 83] only formulate $A$ as a static, unchanged target model while tracking.

Algorithm 5 Target Tracking by Adaptive GMM and Mean-Shift Optimization.

**inputs:** The readings of the initial event sensor set $\mathcal{R}(t = 0)$ and its center location $L(0)$ and the size of $\mathcal{R}(t)$.  
**outputs:** A sequence of estimated target locations $L(t)$ ($t = 1, 2, \ldots$).

1. **Model Initialization:** Fit $A(t = 0)$ using the EM algorithm, the BIC criterion, and the readings $\{x_{i}(0)\}$ of event sensors where $S_{i} \in \mathcal{R}(t = 0)$. Center $\mathcal{R}(t = 1)$ at the initial target location $L(0)$ and set $t = 1$.

2. **Density Evaluation:** For each sensor $S_{i} \in \mathcal{R}(t)$, evaluate the value of the probability density function $p(x_{i}(t) \mid A'(t - 1))$ using Eq. (4.19).

3. **Mean-Shift Optimization for Tracking:** Use the mean-shift iterative process to find the new event location $L(t)$ according to the iteration Eq. (5.1).

4. **Incremental Model Updating:** Update $A(t)$ using the incremental EM algorithm [52], given the new event sensor set $\mathcal{R}(t)$.

5. Wake up the next active sensor set $\mathcal{R}(t + 1)$ centered at the new target location $L(t)$. Set $t = t + 1$, and go to step 2.

5.2.3 Adaptive GMMs for Target Updating

As a general form of data representation, GMM can be adopted for modeling and tracking a moving target over time. However, in some scenario, when the target travels to some area where something significant happens in the background, the target sensing values will be affected, mainly in an additive manner. It means that the sensing signals are the addition of the signals capturing both the background sources and the target source. On the other hand, the outlying sensor readings in the set of the event sensors may distort the target
Figure 5.1: A scenario for target tracking where the white circles represent sleeping nodes, blue circles denote active nodes, and red circles are event sensors that contribute to the target location estimation at each time frame. (a) shows the initial step for model fitting at frame $t - 1$, (b) is the process for mean-shift optimization to estimate $\hat{R}(t)$ at each iterations, and (c) shows the final estimated target location at frame $t$. 

83
appearance model, leading to inaccurate target position estimation. For these reasons, we need an extension to temporally adapt the mixture models to capture the possible changes or dynamics of the event data distribution and prevent drifting. In brief, when the target is moving, its appearance model $A$ will normally become dynamic and will change over time.

We can estimate a GMM $\theta(k, T)$ (representing the target at time $T$) using all data $\{X(\tau)\}$ within a $(w + 1)$-time frame temporal window $\tau \in [T - w, T]$. This is a general temporal-spatial fusion treatment for robustness estimation. To filter out outlying sensor readings, $\{X(\tau)\}$ is updated by pruning some sensing measurements that have extremely low probability density values. For implementation convenience, $(1 - p)$ of the original $\{X(\tau)\}$ that have the largest probability density values will be used to compose the new data set $\{X(\tau)\}$ and contribute to estimate the time-changing target model, where $p$ can be set as the ratio of the outlying sensors in the network and this ratio can be empirically acquired. Given the ratio threshold $p$ and $\{X(\tau)\}$, we can use a quick-sort type technique [17] to filter out outlying sensor readings based on their density values $p(x_i(t) \mid A(t - 1))$, which only requires linear time complexity of $|\{X(\tau)\}|$. That is, we fit a dynamic target appearance model $\theta(k, T)$ at each $T$ using some temporally integrated data. We assume that the number of mixture components $k$ is a constant throughout the tracking process. If more computational power is permitted, model selection techniques can be used at each time frame for more accurate model estimation. For the purpose of only evaluating the density values of $p(x_i(t) \mid A(t - 1))$, a fixed $k$ is found to be sufficient empirically, after the process of model selection using BIC at frame 0 as target model pre-calibration. Thus, we may simplify $\theta(k, T)$ as $\theta(T)$. The model $\theta(T)$ can be estimated using the EM algorithm in the following way. In the E-step, we compute

$$\psi_j^{(\tau)} = \sum_{x \in X(\tau)} \gamma(x, j)$$

where $\gamma(x, j)$ denotes the probability that the j-th component generates $x$ and is estimated
using the Bayes’ theorem and the current parameter values. In the M-step, we compute the updates of the parameters

\[ \mu_j(T) = \frac{\sum_{\tau = T-w}^{T} \sum_{x \in \mathcal{X}(\tau)} \gamma(x, j) x}{\sum_{\tau = T-w}^{T} \psi_j^{(\tau)}} , \]

\[ \Sigma_j(T) = \frac{\sum_{\tau = T-w}^{T} \sum_{x \in \mathcal{X}(\tau)} \gamma(x, j) (x - \mu_j(T))(x - \mu_j(T))'}{\sum_{\tau = T-w}^{T} \psi_j^{(\tau)}} , \]

and

\[ \alpha_j(T) = \frac{\sum_{\tau = T-w}^{T} \psi_j^{(\tau)}}{\sum_{j=1}^{k} \sum_{\tau = T-w}^{T} \psi_j^{(\tau)}} . \]

Based on [52], we implement an incremental version of the EM algorithm to update \( \theta(\tau = T) \) more efficiently by integrating the previous model \( \theta(\tau = T - 1) \) (up to frame \( T - 1 \)) and the mixture model estimated with \( \mathcal{X}(T - w - 1) \) at frame \( T - w - 1 \), or the model estimated with \( \mathcal{X}(T) \) at frame \( T \). Accordingly, we denote these two models by \( \{\mu_j^{(T-w-1)}, \Sigma_j^{(T-w-1)}, \alpha_j^{(T-w-1)}\} \) and \( \{\mu_j^{(T)}, \Sigma_j^{(T)}, \alpha_j^{(T)}\} \), with

\[
\mu_j(T) = \mu_j(T - 1) + \frac{\psi_j^{(T)}}{\Psi_j^{(T)}} (\mu_j^{(T)} - \mu_j(T - 1)) - \frac{\psi_j^{(T-w-1)}}{\Psi_j^{(T)}} (\mu_j^{(T-w-1)} - \mu_j(T - 1)) \tag{5.2}
\]

\[
\Sigma_j(T) = \Sigma_j(T - 1) + \frac{\psi_j^{(T)}}{\Psi_j^{(T)}} (\Sigma_j^{(T)} - \Sigma_j(T - 1)) - \frac{\psi_j^{(T-w-1)}}{\Psi_j^{(T)}} (\Sigma_j^{(T-w-1)} - \Sigma_j(T - 1)) \tag{5.3}
\]
and
\[
\alpha_j(T) = \alpha_j(T - 1) + \frac{1}{w+1}(\alpha_j^{(T)} - \alpha_j(T - 1)) \\
- \frac{1}{w+1}(\alpha_j^{(T-w-1)} - \alpha_j(T - 1))
\]  
\quad (5.4)

where
\[
\Psi_j(T) = \sum_{\tau=T-w}^{T} \psi_j^{(\tau)}
\]

It is easy to obtain
\[
\Psi_j(T) = \Psi_j(T - 1) + \psi_j^{(T)} - \psi_j^{(T-w-1)}
\]  
\quad (5.5)

to update \( \Psi_j(T) \) incrementally. Note that we assume that the total data sample count of \( X(\tau) \) remains unchanged over time. Using the incremental model updating technique requires less computation overhead, communication cost and memory storage.

The above steps are summarized into Algorithm 5 for fault tolerant target tracking by combining adaptive GMM and the mean-shift optimization.

### 5.3 Simulations

In this section, we describe our experimental settings and report our evaluation results on fault tolerant target tracking. Numerical study and analysis of our newly proposed algorithm compared with previous works \cite{7, 43, 83} are also provided.

#### 5.3.1 Simulation Setup

Our simulations are performed within a map consisting of \( 512 \times 512 \) simulated sensors with each randomly deployed in one of the \( 512 \times 512 \) grids, restricting one sensor per grid.
Similar to figure 4.2, we use an image to display the sensor map with multivariate sensing values. We assume that sensors have multivariate sensing values from three different modalities, which form a three-component vector shown in RGB color. The extension of multivariate towards more than three dimensions is straightforward.

The background sensing function is set to be Gaussian noise $\mathcal{G}(\mu_0, \Sigma_0)$, where $\mu_0 = \text{randn}(3, 1) \times 8 + [25, 25, 25]'$ and $\Sigma_0$ is a symmetric, positive semi-definite matrix with diagonal elements as $\sigma$ and other elements as $\text{randn}(1, 1) \times \sigma \times 0.3$ under $\Sigma_0(i,j) = \Sigma_0(j,i)$. Function $\text{randn}(m,n)$ returns a $m \times n$ matrix, of which each matrix element satisfies the standard Normal distribution $\mathcal{N}(0,1)$. Diagonal elements of the covariance matrix represent self-correlation and off-diagonal elements indicate cross-correlation between pairwise multivariate variables. The use of multivariate Gaussian function (in our algorithm and simulations) provides a principled way of modeling and fusing multivariate sensor readings. In all the simulations, we set $\sigma = 5$ by default.

**Background Events:** We also evaluate our algorithm under complex, non-uniform background events. Thus, besides the general background noise model mentioned above, we randomly generate another two ellipse-shaped background event regions (which can be considered as “systematic noises” for our target tracking). Each region has a spatial support in the range of $[\text{Dim}/6, \text{Dim}/3]$ for both axes and has unconstrained orientation, where $\text{Dim} = 512$ is the dimension of the network. Its signal model satisfies an exponentially decaying multivariate Gaussian model $\mathcal{G}(\mu_i, \Sigma_i) \times \exp(-d((x, y), (cx_i, cy_i))/\lambda)$, where each dimension of $\mu_i$ is sampled from $[25, 50, 75, 100]$ independently, and $d((x, y), (cx_i, cy_i))$ is the Euclidian distance between the center of the $i$th background noise region $(cx_i, cy_i)_{i=1,2}$ and the sensing location $(x, y)$. The decay factor $\lambda$ is set as 7. We also introduce an additive Gaussian disturbance of $\mathcal{N}(0, 8)$ to each dimension of $\mu_i$. Examples of these two non-uniform background noise regions are shown as light purple or light green (due to randomness) ellipse-shaped blobs at the first row in Fig. 5.3.

**Target Distribution Modeling:** We simulate two target sensing signal patterns as
follows. The moving target is first simulated as a small overlaying pattern of two Gaussian signals (a Gaussian mixture model), which is spatially bounded by a fitted ellipse (target boundary), denoted by the set of event sensors $\mathcal{R}$, in the model calibration process before tracking. The second target pattern is formulated as a decaying signal $X \times \exp(-d((x, y), (cx, cy))/\lambda)$, where $X$ is the signal strength at the center, and $d((x, y), (cx, cy))$ is the distance from its signal center $(cx, cy)$ to any sensor location $(x, y)$. Note that $X$ is a three dimensional vector with each dimension being randomly sampled from a 7-component vector $[100, 125, 150, 175, 200, 225, 250]$. An additive Gaussian disturbance (noise) $\mathcal{N}(0, 10)$ is also introduced to each dimension of $X$. Also notice that $\lambda$ controls the decaying speed in the spatial domain and is set to be $\lambda = 7$. The resultant target boundary is a circle with a radius $= 2 \times \lambda$ in our simulations. We further formulate the target motion trajectory as a circle with a radius randomly selected from $[Dim/4, Dim/3]$, which is proximately deployed in the center of the map. The target moves at a pace uniformly selected from $[1.5, 2.5]$ as a degree of angular distance at each time frame counterclockwise. The window size is set to be $w + 1 = 5$ for the adaptive mixture model in algorithm 5. The radius of the active sensor set $\mathcal{R}^\prime$ is twice as large as that of the event sensor set $\mathcal{R}$.

**Outlying Sensors**: To explore the robustness of our algorithm against outlier sensors, we require that with a probability less than $\varphi$ each sensor node has an outlying reading in any of its three sensing channels. In our simulation, we first generate a uniformly sampled random variable $\varrho \in [0, 1]$ for each sensor, and then randomly select one of its three sensing modalities (i.e., any dimension $x^j_i$ from its three multivariate readings $x_i$) to add $x^j_i$ with a random disturbance ranging from $[-100, 100]$ by randomness if $\varrho < \varphi$. We vary the value of $\varphi$ from 0.05 to 0.25 in our simulation study.
5.3.2 Target Tracking Results

In this section, we report our simulation results under two scenarios: with outlying sensors and without outlying sensors. In each case, we report the tracking performance and our analysis for two noise models (with “background events” or not) and two target models (i.e., the Gaussian mixture model and the spatially decaying signal model). All results are averaged over 100 random trials.

Tracking Results Without Outlying Sensors

We first evaluate our algorithm for tracking a target whose distributional signals are modeled by the Gaussian mixture model. As a comparison, we also implement a variant of algorithm 5 with a fixed GMM [7, 43, 83]. Under the general background noise model (i.e., without the two background event regions), both algorithm 5 and its fixed GMM variant can successfully track the target through the entire time sequence. Based on the numerical results reported in table 5.1, the location errors with a fixed GMM are slightly higher than those of algorithm 5 with an adaptive GMM though in general the performance difference is not significant.

Fig. 5.2 illustrates an example of tracking a moving target over a circular trajectory trespassing the two “background event” regions in the sensor map. Comparisons of the ground truth (shown with red ♦) versus the tracking trajectories output from Algorithm 5 with an adaptive GMM (shown with blue +) and its fixed GMM variant (shown with green ○) demonstrate the superiority of adaptive GMM on capturing the distribution of temporally changing multivariate signals. These results indicate that our adaptive GMM (algorithm 5) is able to track the moving target over the entire spatial region while the fixed GMM variant fails to track the target after a number of time frames.

Fig. 5.3 illustrates several tracking snapshots by Algorithm 5 at different time frames under
Figure 5.2: An example of tracking a moving target over a trajectory trespassing the two background event regions in the sensor map. Red shows the ground truth (the trajectory of the target), while blue is the tracking trajectory from Algorithm 5 with an adaptive GMM model; and green is associated with the fixed GMM variant. It is obvious that Algorithm 5 can track the moving target over the entire spatial range while the fixed GMM variant fails to track the target after a number of time frames.
the above scenario. It is shown that our Algorithm 5 can accurately track the moving target. The adaptive GMM can successfully capture the dynamic appearance (as sensor readings) distributions of the target and track its spatial trajectory with the mean-shift [16] optimization over a sequence of 180 frames. Fig. 5.4 shows the details at frame 12# and 19# for the fixed GMM variant of algorithm 5. It is observed that the fixed mixture model variant loses its tracking at frame 12# when the target enters the first non-uniform background event region (in light-green color). This failure is no surprise in that it captures the sensor reading data distribution only at frame 0. When the target moves to a background event region (as shown in Fig. 5.4), the underlying sensor reading distribution changes dramatically and therefore the fixed GMM based appearance model becomes invalid.

Table 5.1 reports both the absolute localization errors and the standard deviations of Algorithm 5 and its fixed GMM variant over 100 trails under different settings of noise and target models without outlying sensor impacts. The fixed GMM variant can only track the object trajectory under the general simple Gaussian noise background model, and it fails at which the background event regions appear (as in figure 5.4) regardless of the target models. In addition, the mean location errors with the fixed GMM are slightly higher than those of Algorithm 5 in all cases. On the other hand, our adaptive GMM based algorithm 5 can successfully track the moving target under both background noise models for both target models. The adaptive GMM dynamically updates the sensor reading distributions of the moving target and tracks its spatial trajectory with the mean-shift [16] optimization over a sequence of 180 frames. Therefore, Algorithm 5 performs better than the fixed GMM variant in all scenarios. The difference of the tracking performance under the Gaussian mixture target model and the spatially decaying target model seems not significant. This observation demonstrates the flexibility and effectiveness of our Algorithm 5, especially the adaptive GMM based target appearance modeling, for different target models.
Figure 5.3: Sampled frames of tracking the moving target using our proposed Algorithm 5 in a 180 frame sequence. Target is successfully tracked at both the non-event background area and the two event background regions. The red ellipse shows the target boundary based on the ground truth; the blue ellipse shows the tracked target boundary using Algorithm 5.
Figure 5.4: Sampled frames of tracking the moving target using the fixed GMM variant of Algorithm 5 in a 180 time frame sequence. Note that the fixed GMM variant loses the target at frame 19#.

Figure 5.5: Sampled frames of tracking the moving target using algorithm 5-B when 20% outlying sensors exist. The target is successfully tracked in a 180 frame sequence.
### Table 5.1: Comparison study of Algorithm 5 and its fixed GMM variant under the general background noise model and the complex noise model. The first four lines of numerical results are for the Gaussian mixture target model, and the last four lines are computed using the spatially decaying target model.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>abs</th>
<th>std</th>
<th>general</th>
<th>complex</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x$-coor.</td>
<td>$y$-coor.</td>
<td>$x$-coor.</td>
<td>$y$-coor.</td>
</tr>
<tr>
<td>abs</td>
<td>3.1453</td>
<td>3.1383</td>
<td>3.1453</td>
<td>3.1383</td>
</tr>
<tr>
<td></td>
<td>3.7249</td>
<td>3.7336</td>
<td>3.7249</td>
<td>3.7336</td>
</tr>
<tr>
<td>std</td>
<td>0.4154</td>
<td>0.4028</td>
<td>3.2376</td>
<td>3.3427</td>
</tr>
<tr>
<td></td>
<td>0.4445</td>
<td>0.4420</td>
<td>3.5321</td>
<td>3.4463</td>
</tr>
<tr>
<td>abs</td>
<td>0.7352</td>
<td>0.4639</td>
<td>3.6842</td>
<td>0.6352</td>
</tr>
<tr>
<td></td>
<td>0.5742</td>
<td>0.6463</td>
<td>2.6283</td>
<td>2.5837</td>
</tr>
<tr>
<td>std</td>
<td>0.7352</td>
<td>0.4639</td>
<td>3.6842</td>
<td>0.6352</td>
</tr>
<tr>
<td></td>
<td>0.5742</td>
<td>0.6463</td>
<td>2.6283</td>
<td>2.5837</td>
</tr>
</tbody>
</table>

Tracking Results With Outlying Sensors

In the following, we provide performance evaluations for different outlying sensor ratios $\varphi$ under the two randomly deployed background event noise model and the Gaussian mixture target model using Algorithm 5 which is based on the adaptive GMM for target tracking. The performances of other combinations of noise/target models are quite similar. As described in section 5.2.3, an additional low-density sensor reading filtering process can be added in the step of Incremental Model Updating in Algorithm 5 to deal with outlying sensors. Thus we have two versions of Algorithm 5, denoted by 5-A for the case of without sensor filtering and 5-B for the case of with filtering. In these simulations, we assume that there is no perfect pre-calibration, and $p$ is set to be 10% regardless of the real settings of the ratio $\varphi$.

We run 100 trails under $\varphi = 5\%, 10\%, 15\%, 20\%, 25\%$ using either Algorithm 5-A or 5-B. A tracking is successful if the target is successfully tracked (with the mean location error $< 5$ in both $x$ and $y$ coordinates) throughout the full-length time sequence. The rates of successful tracking are reported in Table 5.2. As shown in this table, the sensor “sorting-filtering” process for updating $A(t)$ in Algorithm 5 noticeably improves the successful tracking rates of 5-B though 5-A also achieves good results. Sample tracking snapshots
Table 5.2: Comparison of Algorithms 5-A and 5-B, using the criterion of rates of success of tracking versus total trials.

<table>
<thead>
<tr>
<th>$\varphi$</th>
<th>5%</th>
<th>10%</th>
<th>15%</th>
<th>20%</th>
<th>25%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algo.-A</td>
<td>100%</td>
<td>99%</td>
<td>97%</td>
<td>90%</td>
<td>85%</td>
</tr>
<tr>
<td>Algo.-B</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>99%</td>
<td>97%</td>
</tr>
</tbody>
</table>

of using Algorithm 5 is illustrated in Fig. 5.5. As shown in Fig. 5.6, Algorithm 5-B alsomarginally improves the tracking accuracy in terms of the averaged location error compared to 5-A (after tracking failure trials being removed). However, there is no clear tendency from the statistics of the averaged standard deviation to prove the superiority of 5-B or 5-A.

These results indicate that our algorithm has a desirably good generality on tracking when $5\% \sim 25\%$ outlying sensors exist. This is probably due to the smoothness offered by the probabilistic nature of GMM (to handle extremely noisy sensor readings generated by outlying sensors) and the continuous optimization of the mean-shift algorithm (to accurately locate the target’s moving trajectory over time).

5.4 Discussions

Our algorithm is distributed in nature and can be executed by the cluster head or the leader node of the set of event sensors at each time frame. The EM algorithm has a linear complexity of $O(md)$ where $m$ is the number of (multivariate) sensor readings $\{x_n\}$, and $d$ is the dimensionality of each $x_n$. The incremental GMM updating scheme (in section 5.2.3) only needs a fraction of the computations resulted from a standard EM optimization, but it requires slightly more memory resource. In the incremental version of the EM, we only need to transmit the parameters of the adaptive mixture model. Thus, the communication cost is $O(k + 2kD)$ (float numbers) for diagonal covariance matrices, or $O(k + kD(D + 1)/2 + kD)$ for full matrices.
Figure 5.6: Plot of mean location errors of Algorithms 5-A and 5-B, based on the average of 100 trails per configuration.
The Gaussian or Normal distribution is the most suitable parametric function approximating the natural signals due to its many statistical properties. This is where “Normal” comes from. It has been widely used in many domains, especially signal processing. Sensor readings are also captures of natural signals to which the Gaussian distribution applies. Furthermore, the Gaussian mixture model (GMM) has the modeling capacity of approximating any arbitrary functions. GMM has been used previously in [7, 43, 83] for information processing in sensor networks.

To perform our algorithm 5 for target tracking, the model estimation utilizes the EM algorithm and the BIC criterion, which includes log and exp operations. These two operations can be approximated using Taylor series expansion. Therefore the computation overhead of our algorithm in real sensor implementation is not prohibitive. We propose to implement our tracking algorithm in a real sensor testbed in our future work.
Chapter 6: Conclusion and Future Work

As we stated above, sensor network has emerged as an important, active research area which not only involves a tremendous amount of research efforts (in problem formulation, hardware and communication protocol design, system deployment and maintenance), but originates many exciting, real world applications in map information service, traffic monitoring, ecological system information collection and analysis, hazard preventions and so on. By following this trend, in-network information processing methods has demonstrated its powerful usage and representative modeling capacity in mathematically analyzing huge amount of data, streaming from sensor networks, and discovering semantics from their spatial and temporal data correlations. As marriage of sensor network and information processing, in this thesis, we present two algorithmic schemes for identifying outlying sensors and detecting event boundary sensor, and target tracking in sensor networks.

We propose and analyze two novel Median algorithms for outlying sensor identification and event boundary detection [22]. 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% 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 works that take only 0/1 decision predicates. Therefore, our algorithms are more generic. They can be applied as long as “event” 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 [20, 57].

We extensively develop another category of novel algorithms [25] for event frontline sensor detection, using on statistical clustering methods with model selection. A Boundary
sensor is considered as associated with a multi-modal local neighborhood of (univariate or multivariate) sensing readings, and each Non-Boundary sensor is treated as with a unimodal sensor reading neighborhood. Furthermore, the set of sensor readings within each sensors spatial neighborhood is mathematically formulated using Gaussian Mixture Models. The binary classes of Boundary and Non-Boundary sensors can be effectively classified using the model selection techniques of finite mixture models [53]. We further propose its temporally adaptive version for dynamic target tracking, under a unified statistical mixture modeling framework. Our mixture model based algorithms can be implemented within each purely localized sensor neighborhood and thus scale well to large sensor networks. Their computational overhead is moderately comparable with our former Median based approaches. Simulation results indicate that our new algorithm can effectively detect the event boundary and simultaneously identify outlying sensors with high accuracy under moderate noise ratios. Desirable target tracking results are also achieved under challenging background conditions. To the best of our knowledge, our Gaussian Mixture Model based approach is the first work that presents a principled statistical methodology integrating multi-modality sensor readings for event boundary detection. It is also generic to form a unified framework for fault tolerant target tracking under static and changing background signal models.

In future work, we intend to study other techniques for thresholds (both $\theta_1$ and $\theta_2$) computation for Median based approaches. For example, $\theta$ 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 [22] and chapter 3, 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 $\chi^2$ analysis to make decisions instead of the thresholds.
To further advocate and enhance our Gaussian mixture model based methods, we plan to investigate the issue of event tracking with the event shape changes, i.e., event shape split, event shape merge using data association techniques, event scale change using explicit scale detection. We will also employ machine learning methods [3] for outlier detection in both temporal and spatial domain, to identify different sensor faults as addressed in [68], in a pattern classification [26] framework. Finally we aim to study how to recognize sensor events from different resources (e.g., a certain type of vehicle) and track the scale changes (besides location) of a moving event region in our future research.
Bibliography


[32] Tian He, Suda Krishnamurthy, John A. Stankovic, Tarek F. Abdelzaher, Liqian Luo, Radu Stoleru, Ting Yan, Lin Gu, Jonathan Hui, and Bruce Krogh, An Energy-Efficient


[65] http://www.cse.iitk.ac.in/users/cs725/lec_notes/senslide.html


Appendix A: Effectiveness constraint based Normality Property

Given a univariate Gaussian distribution \( G(\mu, \sigma^2) \), the probability \( p \) of a data sample \( x \in G(\mu, \sigma^2) \) lying \( \theta \) times of the standard deviation \( \sigma \), away from the mean \( \mu \) is uniquely determined by \( \theta \). From the property of Normal or Gaussian distribution, \( p(|x - \mu| >= \theta \sigma) \) is deceasing rapidly as \( \sigma \) increasing. For instance, \( p = 0.26\% \) when \( \theta = 3.00 \), \( p = 5\% \) when \( \theta = 1.96 \), and \( p = 25\% \) when \( \theta = 1.15 \).

As in 4.2.2, remind that the task is to effectively distinguish random variables \( d_i \in G(0, 2\sigma_1^2) \) and \( d'_i \in G(\mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2) \) or \( G(\mu_2 - \mu_1, \sigma_1^2 + \sigma_2^2) \). We first take \( d'_i \in G(\mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2) \) as an example. Without of loss of generality, we assume \( \mu_1 > \mu_2 \). From Normality property, the effectiveness constraint becomes

\[
(\mu_1 - \mu_2) - \theta \sqrt{\sigma_1^2 + \sigma_2^2} > 0 + \theta \sqrt{2}\sigma_1 \quad \text{(A.1)}
\]

to guarantee the good separability of distributions between \( G(0, 2\sigma_1^2) \) and \( G(\mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2) \). This is equivalent to

\[
(\mu_1 - \mu_2) > \theta(\sqrt{\sigma_1^2 + \sigma_2^2} + \sqrt{2}\sigma_1) \quad \text{(A.2)}
\]

or

\[
(\mu_1 - \mu_2) > 2\sqrt{2}\theta \sigma \quad \text{(A.3)}
\]

when \( \sigma_1 = \sigma_2 = \sigma \). For \( d'_i \in G(\mu_2 - \mu_1, \sigma_1^2 + \sigma_2^2) \), we have

\[
(\mu_2 - \mu_1) + \theta \sqrt{\sigma_1^2 + \sigma_2^2} < 0 - \theta \sqrt{2}\sigma_1 \quad \text{(A.4)}
\]

which is the same as equations A.2, A.3. Considering the validity of both cases \( \mu_1 > \mu_2 \) and \( \mu_1 < \mu_2 \), the final effectiveness constraint is derived as equations 4.13, 4.14 in section 4.2.2, by replacing \( (\mu_1 - \mu_2) \) with \( |\mu_1 - \mu_2| \).
Appendix B: Farthest-point Clustering Algorithm for Initialization of $EM$

The problem of partitioning any given sensor reading set $\{x_i, i = 1, 2, \ldots, m\}$ into $k$ subgroups (Kmean or GMM) is equivalent to the NP-hard $k$-center problem. The definition of $k$-center is as follows: given a data set of $n$ points and a predefined cluster number $k$, find a partition of the points into $k$ subgroups $P_1, P_2, \ldots, P_k$ and the data centers $c_1, c_2, \ldots, c_k$, to minimize the maximum radius of clusters $\max_i \max_{p \in P_i} \| p - c_i \|$, where $i$ is the index of clusters. An efficient greedy algorithm, farthest-point clustering, which proved to give an approximation factor of 2 of the optimum. The algorithm operates as follows: pick a random point $p_1$ as the first cluster center and add it to the center set $C$; for iterations $i = 2, \ldots, k$, find the point $p_i$ with the farthest distance to the current center set $C$: $d_i(p_i, C) = \min_{c \in C} \| p_i - c \|$ and add $p_i$ to set $C$; finally assign data points to its nearest center and recompute the means of clusters in $C$. This algorithm is computationally efficient and theoretically bounded that can be used to replace the random initialization step in $EM$. 

111
Appendix C: Kernel Mean-shift Target Tracking

Algorithm 6 Algorithm: Kernel Mean-shift Target Tracking

\textbf{inputs:} Sensors \( \{s_i, i = 1, 2, \ldots, N\} \); their readings \( \{x_i, i = 1, 2, \ldots, N\} \) and locations \( \{z_i, i = 1, 2, \ldots, N\} \) in the sensor field; Target’s signal profile \( P \) and its initial location \( L_1 \);

\textbf{outputs:} Target’s locations \( L_j, j = 2, \ldots, T \).

1. \( t=1 \); set \( L = L_t \).
2. From \( L \), select a subgroup of sensors \( SS = \{s_i\} \), if \( d(x_i, L) < D \) where \( D \) is a predefined distance range of the target.
3. For every sample \( s_i \in SS \), use a similarity function \( W_i = F(x_i, P) \) to compute the weight or likelihood of target at \( s_i \).
4. Compute \( L' \) as a kernel weighted sum of \( \{z_i\} \) where \( s_i \in SS \).

\[ L' = \frac{\sum_{s_i \in SS} \{z_i W_i \mathcal{G}(\|z_i - L\|^2)\}}{\sum_{s_i \in SS} \{W_i \mathcal{G}(\|z_i - L\|^2)\}} \]  

(C.1)

5. Compare \( L \) and \( L' \), if \( d(L, L') < \epsilon \), converge and \( L_{t+1} = L' \); otherwise, \( L = L' \) and go to (2).
6. \( t=t+1 \); set \( L = L_t \), if \( t=T \), exit; otherwise go to (2).

In practice, kernel with Epanechnikov profile is normally used to simplify equation (6) as

\[ L' = \frac{\sum_{s_i \in SS} \{z_i W_i \}}{\sum_{s_i \in SS} \{W_i \}} \]  

(C.2)

The similarity function design is application-dependent based on the formulation of matching from sensor readings \( x_i \) towards the target signal profile \( P \). This method is directly applicable to sensor readings with multiple attributes; \( x_i \) and \( P \) will be vectors instead of scalars.