# Fault Tolerant Target Tracking in Sensor Networks

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

Xiuzhen Cheng
Dept. of Computer Science
The George Washington University
Washington DC 20052, USA
cheng@gwu.edu

## ABSTRACT

In this paper, we present a Gaussian mixture model based approach to capture the spatial characteristics of any target signal in a sensor network, and further propose a temporally-adaptive variant of the approach for dynamic multiple target tracking under changing environments, with the presence of both significant background event noises and a large portion of outlying sensor readings. The target position is estimated by adopting the mean-shift optimization to discriminate the target signals from the background noises. 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. This consideration is practical as in real world applications, sensor readings are multi-modal and may contain errors. The simulation study validates our design and the results indicate that our mixture model based algorithm is an effective and capable approach for the two most typical target signal models under consideration. Desirable quantitative target tracking results are also achieved through extensive evaluations under challenging background conditions.

## Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Wireless Communication, Information Processing

## General Terms

Algorithms, Design, Reliability, Theory

## Keywords

Target Tracking, Fault Tolerance, Gaussian Mixture Model, Expectation-Maximization, Wireless Sensor Networks

## 1. INTRODUCTION

Sensor networking and the methods for efficiently in-network processing the sensor readings have emerged as important topical thrusts that trigger many real world applications and research activities such as the Microsoft SensorMap [20], Google Earth embedded sensor networks [9], and landslide prediction [22]. 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. Second, sensor readings are prone to noise in real environments and unreliable inter-sensor communications can cause further information loss. Third, each sensor has very limited computational power, memory storage and energy supply, which make computationally expensive algorithms less valuable. These challenges motivate us to design robust, efficient, distributed, and in-network information extraction algorithms.

In this paper, 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 first propose a novel unified solution based on the Gaussian mixture model [6, 8, 18] with an explicit model selection to model the distribution of the target signals. For target tracking, we adopt an adaptive Gaussian mixture representation to handle the target mobility issue and employ the Mean-shift [4, 15] 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 presence of one or more targets.

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 impor-


*MobiHoc'09,* May 18–21, 2009, New Orleans, Louisiana, USA.

tance, but have not been fully addressed in previous work [3, 11, 12, 14, 7, 25].

Extensive simulation results are also reported in this paper, which demonstrate that our proposed algorithm can accurately track the target motion trajectory, including single target tracking with and without outlying sensors and multiple target tracking with trajectory overlaps, 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 [19] and network traffic mining [10, 16].

The remainder of the paper is organized as follows. In section 2, we briefly review the most related work on target tracking. Then an introduction on several important aspects of the *finite Gaussian Mixture Model* is given in section 3. Next our newly proposed target tracking algorithm is described in detail, followed by a comprehensive simulation evaluation in section 5. We also discuss the computational complexity and implementation issues of our algorithm in section 6 and finally conclude this paper in section 7.

## 2. RELATED WORK

Target tracking has emerged as an interesting problem representing a very important class of sensor network applications [3, 11, 12, 14, 25]. In this section, we briefly survey the most related research.

Refs. [3, 14, 25] 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 target's statistical properties. Based on either the probability density values or the (Mahanabolis) distances from the sensors influenced by the target, [3, 14, 25] 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 paper, 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 networks is designed and implemented in [11, 12]. A power management protocol is adopted to set the sensor nodes either in active or in sleep state 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 moving targets under simple and complex background noise models.

The authors in [23] 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 [24] 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.

## 3. MOTIVATION AND MIXTURE MODELS

The mixture model has a wide variety of applications in practice, ranging from spatial data mining [19], to large scale network dataflow monitoring and fault detection in complex network systems [10, 16], and to effective face tracking [17]. In this paper, we will explore the statistical modeling property of the finite mixture model [6, 8], especially the Gaussian mixture model (GMM), and adopt it into the scenario of distributive (sensor network) sensing data processing and mining. We propose to investigate target tracking based on adaptive GMMs for target signal modeling and iterative mean shift optimization for target location estimation.

### 3.1 Finite Mixture Models and Gaussian Mixture Models

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

$$p(x_i \mid \theta) = \sum_{j=1}^{k} \alpha_j p(x_i \mid \theta_j), j = 1, 2, .., k; i = 1, 2, .., m \quad \text{subject to} : \sum_{j=1}^{k} \alpha_j = 1 \tag{1}$$

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 $\theta = \{\alpha_1, \theta_1, \alpha_2, \theta_2, ..., \alpha_k, \theta_k\}$ as the complete set of parameters defining a specific finite mixture model. The objective function of estimating $\theta$ from $\mathbb{X}$ is to maximize the log-likelihood criterion

$$logp(\mathbb{X} \mid \theta) = \sum_{i=1}^{m} log \sum_{j=1}^{k} \alpha_j p(x_i \mid \theta_j). \tag{2}$$

Then the maximum likelihood estimator of $\theta$ is

$$\hat{\theta}_{ML} = \arg\max_{\theta} \{logp(\mathbb{X} \mid \theta)\}. \tag{3}$$

It is well known [6, 8, 18] that $\hat{\theta}_{ML}$ can not be computed analytically from Eq. (3). Instead, the *expectation maximization* (*EM*) algorithm [6] 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 perhaps the most important class of finite mixture densities [8] because the Gaussian function is the most commonly used function in statistics as an approximation to real world signals. GMM offers flexible and comprehensive unsupervised data modeling capacity and is capable of approximating any underlying density distribution in theory. It is formulated by using a Gaussian density $\mathcal{G}(x_i \mid \mu_j, \Sigma_j)$ with its mean vector $\mu_j$ and covariance matrix $\Sigma_j$ to replace the general probability den-

sity function $p(x_i \mid \theta_j)$ in the finite mixture model

$$p(x_i \mid \theta) = \sum_{j=1}^{k} \alpha_j \mathcal{G}(x_i \mid \mu_j, \Sigma_j), \quad (4)$$

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

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

## 3.2 Expectation Maximization Algorithm

The EM algorithm was originally presented for mixture model parameter estimation in Dempster's paper [6] under the incomplete data condition. It was then extensively developed in the last three decades [8, 18] 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 specifically, the mean vectors $\mu_j$, covariance matrices $\Sigma_j$ and prior weights $\alpha_j$ in GMMs.

The EM algorithm includes iterations of two steps as follows [18, 8]. At each iteration $t$, both the E-step and the 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

$$\gamma_{ij}^t = \frac{\alpha_j^{t-1}\mathcal{G}(x_i \mid \mu_j^{t-1}, \Sigma_j^{t-1})}{\sum_{j=1}^{k} \alpha_j^{t-1}\mathcal{G}(x_i \mid \mu_j^{t-1}, \Sigma_j^{t-1})} \quad (5)$$

*M-step:* Compute the parameters using $\gamma_{ij}^t$

$$\alpha_j^t = \frac{\sum_{i=1}^{m} \gamma_{ij}^t}{\sum_{j=1}^{k}\sum_{i=1}^{m} \gamma_{ij}^t} \quad (6)$$

$$\mu_j^t = \frac{\sum_{i=1}^{m} \gamma_{ij}^t x_i}{\sum_{i=1}^{m} \gamma_{ij}^t} \quad (7)$$

$$\Sigma_j^t = \frac{\sum_{i=1}^{m} \gamma_{ij}^t (x_i - \mu_j^t)(x_i - \mu_j^t)'}{\sum_{i=1}^{m} \gamma_{ij}^t}. \quad (8)$$

The *EM* algorithm generates a sequence of model parameter estimates $\hat{\theta}_{ML}^t, t = 1, 2, ..,$ 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 $\mu_j = random(\{x_1, x_2, \cdots, x_m\})$ 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 [18].

## 3.3 Model Selection

The *EM* algorithm does not provide any information regarding the selection of the number of mixtures from the data. Clearly, such a selection is an important and unavoidable computational issue for GMM and EM. 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] and Bayesian information criterion (BIC) [21] 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 $\mathbb{X}$ as data, AIC is defined

$$AIC(\theta) = -2log(p(\mathbb{X}|\theta)) + 2\mathbb{K} \quad (9)$$

where $\mathbb{K}$ refers to the total number of parameters to be estimated in GMM.

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 $\mathbb{K}$ [13]. On the other hand, the model complexity term of BIC is better formulated as $\mathbb{K}log(m)$ by considering both $\mathbb{K}$ and $m$, compared with $2\mathbb{K}$ in AIC. Thus in this paper, we use the following BIC [21]

$$BIC(\theta) = -2log(p(\mathbb{X}|\theta)) + \mathbb{K}log(m) \quad (10)$$

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. This is equivalent to achieve the minimum of Eq. (10).

There is another body of work on finite mixture models [8] that are capable of selecting the number of mixture modes $k$ during *EM* optimization. For details, we refer the interested readers to [8]. We leave this issue for future work.

# 4. ALGORITHM

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 [14, 25] 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 temporal appearance of the target (in terms of sensing data distribution) and reflects their changes more effectively than the original "calibrated-then-fixed" GMM approach in [14, 25]. To locate the target, the mean-shift algorithm [4, 15] 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 [4].

In summary, this paper tackles the target tracking problem 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 [4, 15] to find the new target location.

## 4.1 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[1] due to hardware failures or harsh environment. Suppose that the outlying readings are uncorrelated in the network.

To save network energy, we assume some power management protocols [11] 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 [18, 8]. Suppose the target signals can be modeled as the weighted sum of some Gaussian functions, ie., a Gaussian mixture model as Eq. (4). 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 3.2 and 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.

## 4.2 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 $\mathcal{L}(t-1)$ denote the location of target event at time $(t-1)$. $\mathcal{R}(t-1)$ is the set of event sensors that will be used to update model $\mathcal{A}(t-1)$ at time $(t-1)$. $\mathcal{R}(t-1)$ is geometrically reconstructed as a sensor neighborhood set around $\mathcal{L}(t-1)$. For simplicity, we assume that all sensors in $\mathcal{R}(t-1)$ reside in a disk centered at $\mathcal{L}(t-1)$, denoted by $\mathcal{D}$. Or, we can intuitively consider $\mathcal{R}(t-1)$ as a "disk" centered at $\mathcal{L}(t-1)$. We further assume that the ship and size of $\mathcal{D}$ 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 $\mathcal{D}$ (ie., expanding $\mathcal{D}$ by two times in both $x$ and $y$ coordinates), but has the same center as $\mathcal{D}$. Denote the set of *active nodes* for target tracking at time frame $t$ as $\mathcal{RR}(t)$. 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 $\mathcal{RR}(t)$ according to the target moving velocity and direction is left as future work. Figure 1 (a) shows a scenario to illustrate the set of event sensors $\mathcal{R}(t-1)$ and the set of active sensors $\mathcal{RR}(t)$ 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 $\mathcal{RR}(t)$ 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 $\mathcal{A}(t-1)$ and the mean-shift algorithm [4]. 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 $\mathcal{A}(t-1)$, we compute the value of the fitness by evaluating the probability density functions of the target appearance model $\mathcal{A}(t-1)$ for the reading $x_i(t)$ of each sensor $S_i$ in $\mathcal{RR}(t)$. These values, denoted by $\{p(x_i(t) \mid \mathcal{A}(t-1))\}$ and appearing as a target-conditional response map (conditioned on $\mathcal{A}(t-1)$), 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

$$\hat{\mathcal{L}}(t,(it+1)) = \frac{\sum_{S_i \in \hat{\mathcal{R}}(t,it)} p(x_i(t) \mid \mathcal{A}(t-1)) L_i}{\sum_{S_i \in \hat{\mathcal{R}}(t,it)} p(x_i(t) \mid \mathcal{A}(t-1))}, \quad (11)$$

where $L_i$ denotes the location of sensor $S_i \in \hat{\mathcal{R}}(t,it)$. Here $\hat{\mathcal{R}}(t,it)$ is the set of "event" sensors that contribute to the

---

**Algorithm 1** 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 $\mathcal{RR}(t)$.
**Outputs:** A sequence of estimated target locations $L(t)$ $(t=1,2,...)$.

1. *Model Initialization:* Fit $\mathcal{A}(t=0)$ 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 $\mathcal{RR}(t=1)$ at the initial target location $L(0)$ and set $t=1$.
2. *Density Evaluation:* For each sensor $S_i \in \mathcal{RR}(t)$, evaluate the value of the probability density function $p(x_i(t) \mid \mathcal{A}'(t-1))$ using Eq. (4).
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. (11).
4. *Incremental Model Updating:* Update $\mathcal{A}(t)$ using the incremental *EM* algorithm [17], given the new event sensor set $\mathcal{R}(t)$.
5. Wake up the next active sensor set $\mathcal{RR}(t+1)$ centered at the new target location $L(t)$. Set $t=t+1$, and go to step 2.

---

[1] A sensor's reading is outlying if it deviates significantly from other readings of neighboring sensors [2].

newly estimated target location at each iteration $it$. $\hat{\mathcal{R}}(t,(it+1))$ is then updated using $\hat{\mathcal{L}}(t,(it+1))$, in the same way as obtaining $\mathcal{R}(t-1)$ from $\mathcal{L}(t-1)$, or simply, by "moving" $\hat{\mathcal{R}}(t,(it))$ to the new location of $\hat{\mathcal{L}}(t,(it+1))$. Note that $\hat{\mathcal{L}}(t,(it+1))$ is the estimated target location, which is the weighted "average" of the locations of the current set of event sensors in $S_i \in \hat{\mathcal{R}}(t,it)$ given $p(x_i(t) \mid \mathcal{A}(t-1))$ from previous step. At the beginning of the iterative process, $\hat{\mathcal{R}}(t,1)$ is the same as $R(t-1)$. For the subsequent iterations, $\hat{\mathcal{R}}(t,(it+1))$ and sensors $S_i \in \hat{\mathcal{R}}(t,(it+1))$ are updated using $\hat{\mathcal{L}}(t,(it+1))$ as follows. We simply choose the active sensor nodes reside in the covering range of $\hat{\mathcal{L}}(t,(it+1))$ as event sensors. When convergent, the final $\hat{\mathcal{L}}(t,(it+1))$ is used to estimate the location $L(t)$. Empirically, the mean-shift converges within $3 \sim 5$ iterations in our experimental settings. Figure 1 illustrates the process of the mean-shift optimization for target tracking and localization. In contrast to our approach, [3, 14, 25] only formulate $\mathcal{A}$ as a static, unchanged target model while tracking.

## 4.3 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 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 $\mathcal{A}$ 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 $\{\mathbb{X}(\tau)\}$ 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, $\{\mathbb{X}(\tau)\}$ is updated by pruning some sensing measurements that have extremely low probability density values. For implementation convenience, $(1-p)$ of the original $\{\mathbb{X}(\tau)\}$ that have the largest probability density values will be used to compose the new data set $\{\mathbb{X}(\tau)\}$ and contribute to estimating 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 $\{\mathbb{X}(\tau)\}$, we can use a quick-sort type technique [5] to filter out outlying sensor readings based on their density values $p(x_i(t) \mid \mathcal{A}(t-1))$, which only requires linear time complexity of $|\{\mathbb{X}(\tau)\}|$. That is, we fit a dynamic target appearance model $\theta(k,T)$ at each $T$ using some temporally integrated data. In this paper, 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 $p(x_i(t) \mid \mathcal{A}(t-1))$, 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

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

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

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

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

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 [17], 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 $\mathbb{X}(T-w-1)$ at frame $T-w-1$, or the model estimated with $\mathbb{X}(T)$ 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

$$\begin{aligned} \mu_j(T) = \mu_j(T-1) &+ \frac{\psi_j^{(T)}}{\Psi_j(T)}(\mu_j^{(T)} - \mu_j(T-1)) \\ &- \frac{\psi_j^{(T-w-1)}}{\Psi_j(T)}(\mu_j^{(T-w-1)} - \mu_j(T-1)) \end{aligned} \tag{12}$$

$$\begin{aligned} \Sigma_j(T) = \Sigma_j(T-1) &+ \frac{\psi_j^{(T)}}{\Psi_j(T)}(\Sigma_j^{(T)} - \Sigma_j(T-1)) \\ &- \frac{\psi_j^{(T-w-1)}}{\Psi_j(T)}(\Sigma_j^{(T-w-1)} - \Sigma_j(T-1)) \end{aligned} \tag{13}$$

and

$$\begin{aligned} \alpha_j(T) = \alpha_j(T-1) &+ \frac{1}{w+1}(\alpha_j^{(T)} - \alpha_j(T-1)) \\ &- \frac{1}{w+1}(\alpha_j^{(T-w-1)} - \alpha_j(T-1)) \end{aligned} \tag{14}$$

where

$$\Psi_j(T) = \sum_{\tau=T-w}^{T} \psi_j^{(\tau)}$$

It is easy to obtain

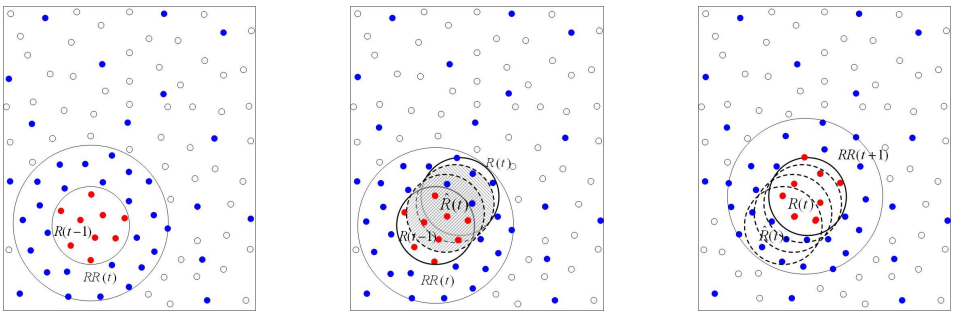
$$\Psi_j(T) = \Psi_j(T-1) + \psi_j^{(T)} - \psi_j^{(T-w-1)} \tag{15}$$

to update $\Psi_j(T)$ incrementally. Note that we assume that the total data sample count of $\mathbb{X}(\tau)$ 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 1 for fault tolerant target tracking by combining adaptive GMM and the mean-shift optimization.

# 5. 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 [3, 14, 25] are also




(a) (b) (c)

**Figure 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 $\hat{\mathcal{R}}(t)$ at each iterations, and (c) shows the final estimated target location at frame $t$.**

provided.

## 5.1 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. 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 Fig. 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 more than three dimensions is straightforward.

The background sensing function is set to be Gaussian noise $\mathcal{G}(\mu_0, \Sigma_0)$, where $\mu_0 = randn(3,1) \times 8 + [25, 25, 25]'$ and $\Sigma_0$ is a symmetric, positive semi-definite matrix with diagonal elements as $\sigma$ and other elements as $randn(1,1) \times \sigma \times 0.3$ under $\Sigma_0(i,j) = \Sigma_0(j,i)$. Function $randn(m,n)$ 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 background noise with 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 $[Dim/6, Dim/3]$ for both axes and has unconstrained orientation, where the dimension of the network 512 is set to 512. 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 $(cx_i, cy_i)$ for $i \in \{1,2\}$ 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. 4.

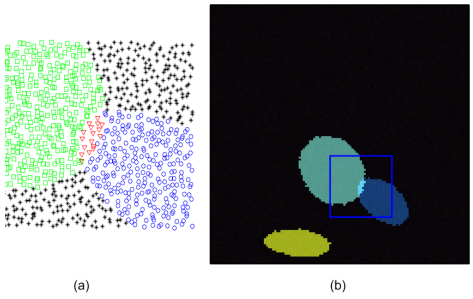
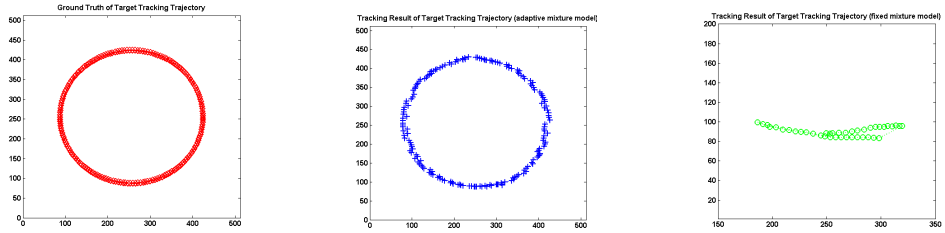



**Figure 2: Sensor map represented using an image. (a) a local sensor map of two sensing events shown in different colors with their spatial locations, (b) a global sensor map represented by an image.**

**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 se-




(a) (b) (c)

**Figure 3: An example of tracking a moving target over a trajectory trespassing the two background event regions in the sensor map. (a) shows the ground truth (the trajectory of the target), while (b) is the tracking trajectory from Algorithm 1 with an adaptive *GMM model*; and (c) is associated with the fixed *GMM* variant. It is obvious that Algorithm 1 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.**

lected 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 1. The radius of the active sensor set $\mathcal{RR}$ 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 (ie., any dimension $x_i^j$ from its three *multivariate* readings $x_i$) to add $x_i^j$ with a random disturbance ranging from $[-100, 100]$ if $\varrho < \varphi$. We vary the value of $\varphi$ from 0.05 to 0.25 in our simulation study.

## 5.2 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 (ie., the Gaussian mixture model and the spatially decaying signal model). All results are averaged over 100 random trials.

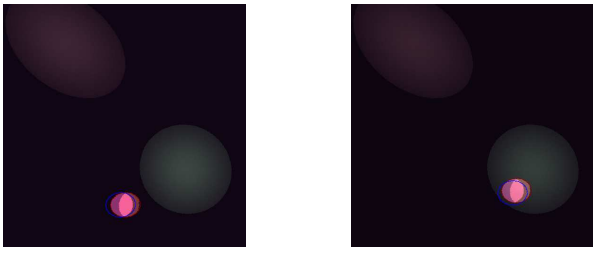
### 5.2.1 *Tracking Results Without Outlying Sensors*

We first evaluate our algorithm for tracking a target whose distributional signals are modeled by the Gaussian mixture model in complex background noise settings. As a comparison, we also implement a variant of algorithm 1 with a fixed GMM [3, 14, 25].

Fig. 3 illustrates an example of tracking a moving target over a circular trajectory trespassing the two "background event" regions in the sensor map. These results indicate that our *adaptive GMM* 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. 4 illustrates several tracking snapshots by Algorithm 1 at different time frames under the above scenario. It is shown that our Algorithm 1 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 optimization [4] over a sequence of 180 frames. Fig. 5 shows the details at frame 12# and 19# for the *fixed* GMM variant of algorithm 1. 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), the underlying sensor reading distribution $\mathcal{A}$ changes dramatically and therefore the fixed GMM based appearance model becomes invalid.

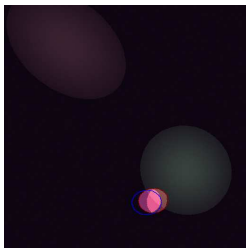
We also compare our algorithm with the fixed GMM [3, 14, 25] in general noise background (without background



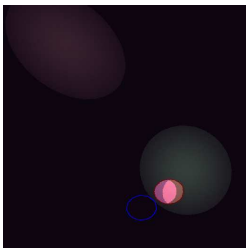
(a) Frame 2# (b) Frame 19#

(c) Frame 99# (d) Frame 110#

**Figure 4: Sampled frames of tracking the moving target using our proposed Algorithm 1 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 1.**



(a) Frame 12#



(b) Frame 19#

**Figure 5: Sampled frames of tracking the moving target using the fixed GMM variant of Algorithm 1 in a 180 time frame sequence. Note that the fixed GMM variant loses the target at frame 19#.**

| Comparison of Algorithms with Adaptive or Fixed GMM | | | | | |
|---|---|---|---|---|---|
| Background | | general | | complex | |
| Algorithm | | Algo.1 | fixed | Algo. 1 | fixed |
| abs | $x$-coor. | 3.1453 | 3.7249 | 3.5420 | — |
| | $y$-coor. | 3.1383 | 3.7336 | 3.3278 | — |
| std | $x$-coor. | 0.4154 | 0.4445 | 2.8422 | — |
| | $y$-coor. | 0.4028 | 0.4420 | 2.9816 | — |

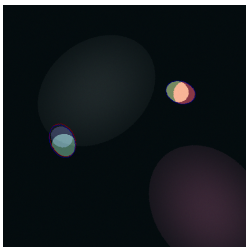
**Table 1: Comparison study of Algorithm 1 and its fixed GMM variant under the general background noise model and the complex noise model for the Gaussian mixture target signals.**

events). Table 1 reports both the absolute localization errors and the standard deviations of Algorithm 1 and its fixed GMM variant over 100 trails under different settings of noise without outlying sensor impacts. Furthermore, the performance result for tracking a target with a decaying signal model is summarized in Table 2. It is shown that 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 Fig. 5) regardless of the target models. In addition, the mean location errors with the fixed GMM are slightly higher than those of Algorithm 1 in all cases. On the other hand, our adaptive GMM based algorithm 1 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 [4] optimization over a sequence of 180 frames. Therefore, Algorithm 1 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 1, especially the adaptive GMM based target appearance modeling, for different target models.*

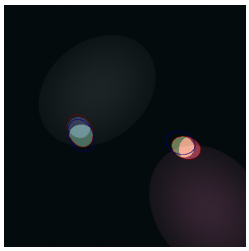
To show the robustness of Algorithm 1 for multi-target tracking, We first perform tracking two targets on different motion patterns. Example tracking frames are demonstrated in Fig. 6. From Fig. 7, one target appears to move along a circle trajectory as before, and the other target moves on a line with a random pace. It is evident that

| Comparison of Algorithms with Adaptive or Fixed GMM | | | | | |
|---|---|---|---|---|---|
| Background | | general | | complex | |
| Algorithm | | Algo.1 | fixed | Algo. 1 | fixed |
| abs | $x$-coor. | 3.2376 | 3.5321 | 3.2376 | — |
| | $y$-coor. | 3.3427 | 3.4463 | 3.1158 | — |
| std | $x$-coor. | 0.7352 | 0.5742 | 2.6283 | — |
| | $y$-coor. | 0.4639 | 0.6352 | 2.5837 | — |

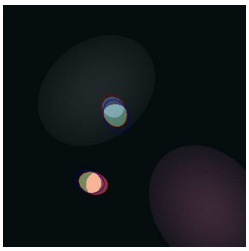
**Table 2: Comparison study of Algorithm 1 and its fixed GMM variant under the general background noise model and the complex noise model for spatially decaying target signal distribution.**



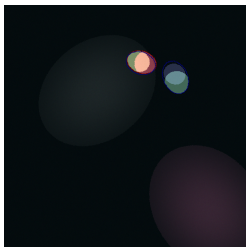
(a) Frame 23#



(b) Frame 48#



(c) Frame 97#



(d) Frame 180#

**Figure 6: Sampled frames of tracking two moving targets, under different motion patterns, by Algorithm 1 in a 180 frame sequence.**

two targets are successfully tracked by Algorithm 1 during the whole process. Handling two moving targets, with overlapping trajectories (in some frames), is also addressed. As shown in Fig. 8, it is expected that the tracked target contours (in blue) drift to a certain degree when the two targets intersect (Frames 48# and 52#) because sensor readings in the overlapped zone do not well satisfy either target GMM model estimated before the targets intersect. However our adaptive GMM based target tracking is capable of robustly recover the lost of tracking and then keeping good tracking accuracy after passing the intersection (Frame 57#). No noticeable tracking accuracy performance degrading is observed for single or multiple target tracking.

### 5.2.2 *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 models and the Gaussian mixture target model using Algorithm 1 based on the adaptive GMM for target tracking. The performances of other combinations of noise/target models are quite similar. As

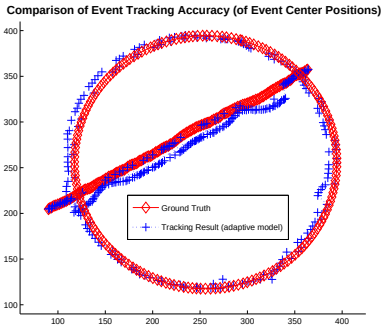



**Figure 7: Ground truth (red) and tracked (blue) target moving trajectories of two targets. One is on a circle-like motion, and the other has line-like motion.**

| $\varphi =$ | 5% | 10% | 15% | 20% | 25% |
|---|---|---|---|---|---|
| Algo.-A | 100% | 99% | 97% | 90% | 85% |
| Algo.-B | 100% | 100% | 100% | 99% | 97% |

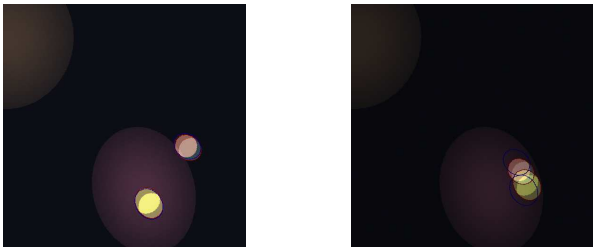
**Table 3: Comparison of Algorithms 1-A and 1-B, using the criterion of rates of success of tracking versus total trials.**

described in section 4.3, an additional low-density sensor reading filtering process can be added in the step of *Incremental Model Updating* in Algorithm 1 to deal with outlying sensors. Thus we have two versions of Algorithm 1, denoted by **1-A** for the case of without sensor filtering and **1-B** for the case of with filtering. In these simulations, we assume that there is no perfect pre-calibration, and the filtering-out ratio $p$ is set to be 10% regardless of the real settings of the ratio $\varphi$.

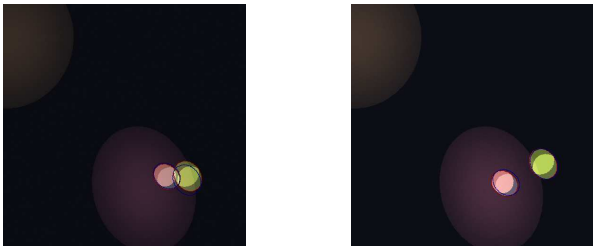
We run 100 trails under $\varphi = 5\%, 10\%, 15\%, 20\%, 25\%$ using either Algorithm **1-A** or **1-B**. A tracking is successful if the target is successfully tracked (with the mean location error $< 5$ in both $x$ and $y$ coordinates) throughout the full-length time sequence. The rates of successful tracking are reported in Table 3. As shown in this table, the sensor "sorting-filtering" process for updating $\mathcal{A}(t)$ in Algorithm 1 noticeably improves the successful tracking rates of **1-B** though **1-A** also achieves good results. Sample tracking snapshots of using Algorithm 1 is illustrated in Fig. 9. As shown in Fig. 10, Algorithm **1-B** also marginally improves the tracking accuracy in terms of the averaged location error compared to **1-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 **1-B** or **1-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).

## 6. DISCUSSIONS

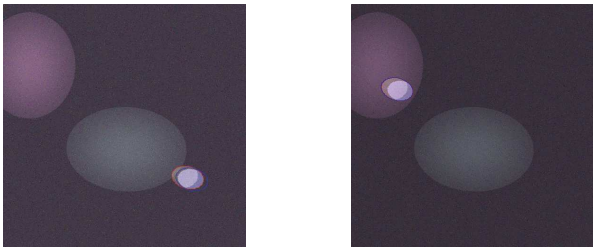


(a) Frame 36# (b) Frame 48#



(c) Frame 52# (d) Frame 57#

**Figure 8: Sampled tracking frames of two overlapping moving targets using Algorithm 1 in another sequence.**



(a) Frame 39# (b) Frame 136#

**Figure 9: Sampled frames of tracking the moving target using algorithm 1-B when 20% outlying sensors exist. The target is successfully tracked in a 180 frame sequence.**

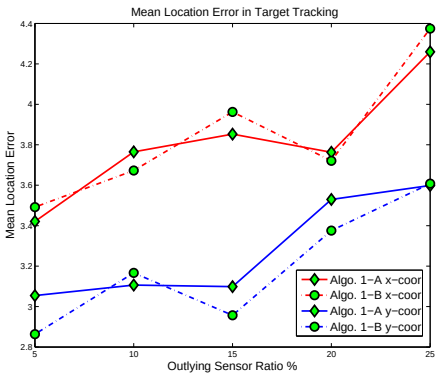



**Figure 10: Plot of mean location errors of Algorithms 1-A and 1-B, based on the average of 100 trails per configuration.**

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 4.3) 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.

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 [3, 14, 25] for information processing in sensor networks.

To perform our algorithm 1 for target tracking, the model estimation utilizes the EM algorithm and the BIC criterion, which includes log and exp operations than 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.

# 7. CONCLUSION

In this paper, we propose a novel unified solution based on the Gaussian mixture model with an explicit model selection to model the distribution of different target signal models. For target tracking, we adopt an adaptive Gaussian mixture representation to tackle the problem of target mobility and employ the continuous optimization of the mean-shift [4, 15] for target localization, with no need of hard classification processes such as in [3, 14] for better tracking accuracy, smoothness and robustness. As a future research, we plan to study how to recognize targets from different sources (e.g., a certain type of vehicle) and track the scale changes (besides location) of a moving event region.

# 8. ACKNOWLEDGMENTS

This research was partially supported by the US National Science Foundation under grants CNS-0347674, CNS-0721669, and CNS-0831852.

# 9. REFERENCES

[1] H. Akaike, Information theory and an extension of the maximum likelihood principle, *2nd International Symposium on Information Theory*, 1973.

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

[3] RR Brooks, P. Ramanathan and AM Sayeed, Distributed target classification and tracking in sensor networks, *Proc. of the IEEE*, 91:1163-1171, 2003.

[4] D. Comaniciu, V. Ramesh, P. Meer, Kernel-Based Object Tracking, *IEEE Trans. PAMI*, 2003.

[5] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, MIT publisher, 2001.

[6] A. Dempster, N. Laird, and D. Rubin, Maximum likelihood estimation from incomplete data via the EM algorithm, *J. of Royal Statistical Society*, 1977.

[7] M. Ding, D. Chen, K. Xing, and X. Cheng, Localized Fault-Tolerant Event Boundary Detection in Sensor Networks, *IEEE INFOCOM*, 2005.

[8] M. Figueiredo and A.K. Jain, Unsupervised learning of finite mixture models, *IEEE Trans. on Pat. Anal. and Mach. Intell.*, 24(3):381-396, 2002.

[9] Research Project: Networks and Infrastructure for Environmental Sensing and Image Acquisition, http://research.cens.ucla.edu/projects/2007/Terrestrial /Systems/

[10] Z. Guo, G. Jiang, H. Chen, and K. Yoshihira, Tracking Probabilistic Correlation of Monitoring Data for Fault Detection in Complex Systems, *Int. Conf. on Dependable Systems and Networks*, 2006.

[11] T. He, et al., An Energy-Efficient Surveillance System Using Wireless Sensor Networks. *Mobisys*, 2004.

[12] T. He, et al., Achieving Real-Time Target Tracking Using Wireless Sensor Networks, *ACM Trans. on Embedded Computing System*, 2007.

[13] C. Hurvich and C. Tsai, Regression and Time Series Model Selection in Small Samples, *Biometrika* 76:297-307, 1989.

[14] D. Li, K. Wong, YH Hu and A. Sayeed, Detection, classification and tracking of targets in distributed sensor networks, *IEEE Signal Processing Magazine*, March 2002.

[15] L. Lu and G. Hager, A Nonparametric Treatment on Location/Segmentation Based Visual Tracking, *IEEE Conf. on Comp. Vis. and Pat. Recog.*, 2007.

[16] X. Meng, G. Jiang, H. Zhang, H. Chen, K. Yoshihira, Automatic Profiling of Network Event Sequences: Algorithm and Applications, *IEEE INFOCOM*, 2008.

[17] S. McKenna, Y. Raja and S. Gong, Tracking colour objects using adaptive mixture models *Image and Vision Computing*, 17(3-4):225-231, 1999.

[18] G. McLachlan and D. Peel, *Finite Mixture Models*. New York: John wiley & Sons, 2000.

[19] R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. *VLDB*, 1994.

[20] http://atom.research.microsoft.com/sensormap/

[21] G. Schwarz, Estimating the dimension of a model, *Annals of Statistics*, Vol. 6, pp. 461-464, 1978.

[22] A. Terzis, A. Anandarajah, K. Moore. I-J. Wang, Slip Surface Localization in Wireless Sensor Networks for Landslide Prediction, *IPSN*, 2006.

[23] X. Yu, K. Niyogi, S. Mehrotra, N. Venkatasubramanian, Adaptive Target Tracking in Sensor Networks, *CNDS*, 2004.

[24] W. Zhang and G. Cao, Optimizing Tree Reconfigration for Mobile Target Tracking in Sensor Networks, *IEEE INFOCOM*, 2004.

[25] F. Zhao, J. Shin and J. Reich, Information-driven dynamic sensor collaboration for tracking applications, *IEEE Signal Processing Magazine*, 2002.