Mining Lines in the Sand: On Trajectory Discovery From Untrustworthy Data in Cyber-Physical System

Lu-An Tang[1], Xiao Yu[1], Quanquan Gu[1], Jiawei Han[1], Alice Leung[2], Thomas La Porta[3]
[1]Dept. of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA;
[2]BBN Technology, Cambridge, MA, USA
[3]Dept. of Computer Science, Pennsylvania State University, University Park, PA, USA
{tang18, xiaoyu1, qgu3, hanj}@illinois.edu, aleung@bbn.com, tlp@cse.psu.edu

ABSTRACT

A Cyber-Physical System (CPS) integrates physical (*i.e.*, sensor) devices with cyber (*i.e.*, informational) components to form a context sensitive system that responds intelligently to dynamic changes in real-world situations. The CPS has wide applications in scenarios such as environment monitoring, battlefield surveillance and traffic control. One key research problem of CPS is called "*mining lines in the sand*". With a large number of sensors (sand) deployed in a designated area, the CPS is required to discover all the trajectories (lines) of passing intruders in real time. There are two crucial challenges that need to be addressed: (1) the collected sensor data are not trustworthy; (2) the intruders do not send out any identification information. The system needs to distinguish multiple intruders and track their movements. In this study, we propose a method called *LiSM* (Line-in-the-Sand Miner) to discover trajectories from untrustworthy sensor data. *LiSM* constructs a watching network from sensor data and computes the locations of intruder appearances based on the link information of the network. The system retrieves a *cone-model* from the historical trajectories and tracks multiple intruders based on this model. Finally the system validates the mining results and updates the sensor's reliability in a feedback process. Extensive experiments on big datasets demonstrate the feasibility and applicability of the proposed methods.

Categories and Subject Descriptors

H.2 [**Database Applications**]: Data Mining

Keywords

Cyber-physical system, sensor network, trajectory

1. INTRODUCTION

A Cyber-Physical System (CPS) is an integration of sensor networks with informational devices [2]. The CPS employs a large number of low-cost, densely-deployed sensors to watch over designated areas and automatically discover passing intruders. Such a system has many promising applications in both military and civilian fields, including missile defense [4], battlefield awareness [3, 15], traffic control [9, 17], neighborhood watch [6], environment monitoring [16] and wildlife tracking [7]. The key problem in the above applications is called "mining lines in the sand" [1], *i.e.*, discovering the trajectories of passing intruders from the collected sensor data.

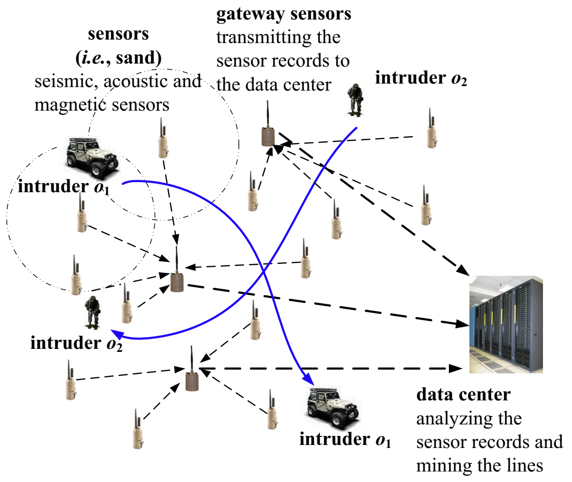

sensors (*i.e.*, sand)
seismic, acoustic and magnetic sensors
gateway sensors transmitting the sensor records to the data center
intruder o_2
intruder o_1
intruder o_2
data center analyzing the sensor records and mining the lines
intruder o_1

Figure 1: The Framework of Battlefield CPS

Figure 1 shows the framework of a battlefield CPS: The sand (seismic, acoustic and magnetic sensors) is deployed in a designated area. It constantly collects signals of vibration, sound and magnetic force from the environment. When an intruder passes by, the sensors detect a signal change and send out detection records. The system analyzes the collected data and discovers the intruder trajectories in real time. Such a system can help military forces see through the "fog of war" and protect troops and bases on the battlefield.

However, the topic of "mining lines in the sand" is considered one of the major challenges in CPS research field, partly due to the following problems:

- Untrustworthy data: Many deployment experiences have shown that untrustworthy (*i.e.*, faulty) data is the most serious problem that impacts CPS performance [13, 16]. Untrustworthy data are generated due to various reasons, including hardware failure, communication limits, environmental influences and so on. It is difficult to filter them out solely based on the signal value, because the values of faulty signals are similar to correct ones.

- Tracking intruders: There are usually multiple intruders in the monitoring area and the system is required to track all of them. Since the intruders do not send out any identification information, the system has to distinguish them and track their movements. Many previous methods make the assumption of single intruder and cannot discover multiple ones in real applications.
- Big data: A CPS usually contains hundreds, even thousands of sensors [2]. Each sensor generates a data record every few minutes; such records form a big dataset. In several applications, actions must be taken immediately to deal with the intruders. The system is required to discover trajectories in real time.

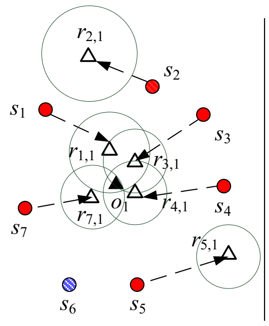
In this study, we propose a novel system called *LiSM* (Line-in-the-Sand Miner) to discover intruder trajectories from untrustworthy sensor data. *LiSM* first constructs a watching network to model the relationship among the sensors, data records and intruders. Then *LiSM* detects the intruder's appearances based on the link information of the watching network. To track multiple intruders, a *cone model* is proposed to generate the intruder trajectories. The system employs a validation process to filter out false positives and updates sensor reliability scores. The technical contributions of this study are summarized as follows.

- We construct a watching network to model the relationship among the sensors, records and intruders. Such a network helps locate the intruder appearances in every timestamp.
- We propose a cone model to track multiple intruders. The cone model is an effective tool to generate trajectories from the detected intruder appearances.
- The system validates the candidate results and filters out false positives. Then the system updates the sensor reliability scores in a feedback process.
- We conduct extensive experiments to evaluate the effectiveness and efficiency of proposed methods on big datasets. The experiment results show that our approach yields higher precision and recall than existing methods.

The rest of the paper is organized as follows. Section 2 introduces the background knowledge and problem formulation; Section 3 proposes the techniques of constructing the watching network, Section 4 introduces the trajectory mining methods; Section 5 conducts the performance evaluations; Section 6 briefly comments on related work; and Section 7 concludes the paper.

2. PROBLEM STATEMENT

Recent advances in sensor technology have produced many types of sensors for area-watching purposes. Such sensors can be roughly classified into two categories by mechanism: (1) Active sensor (*e.g.*, the infrared sensors and radar sensors): these sensors radiate signal pulses and detect objects by the echo bouncing off the intruders; (2) Passive sensor (*e.g.*, the acoustic sensors, seismic sensors and magnetic sensors): these sensors only receive signals from the environment. Active sensors achieve higher accuracy, but require significant more power to operate and drain batteries quickly. Furthermore, when active sensors radiate signal pulses, they are at high risk of being detected by the intruders. As a result, the CPS is usually deployed with a large number of low-cost, energy-saving passive sensors.



intruder	position
o_1	(100, 100)

record	center	radius
$r_{1,1}$	(97,109)	10
$r_{2,1}$	(95,123)	10
$r_{3,1}$	(105,107)	7
$r_{4,1}$	(105,99)	8
$r_{5,1}$	(125,84)	7
$r_{7,1}$	(92,98)	6

Figure 2: Example: The Detection Records

Passive sensors constantly collect signals of sound, vibration and magnetic forces from the environment. When an intruder passes by, the sensors detect it based on the signal changes. However, due to hardware limitation, the sensors can only report the area of an intruder's possible appearance, rather than a concrete point location. In this study, we model the reported area as a planar region bounded by a circle.

Definition 1. (Detection Record) Let s_i be a sensor and t_j be a timestamp, the detection record $r_{i,j}$ is generated by s_i to indicate the possible area of an intruder's appearance in t_j. $r_{i,j}$=$\{cen(r_{i,j}), rad(r_{i,j})\}$, where $cen(r_{i,j})$ and $rad(r_{i,j})$ are the center and radius of the area.

Example 1. Figure 2 shows a list of detection records in time t_1. The solid triangle node is the intruder o_1. The round nodes are nearby sensors watching the area where o_1 passes. The solid round nodes (red) are the responding sensors that send out detection records, such as s_1, s_2 and s_7. The centers of the estimated regions are tagged as hollow triangles. Sensor s_6 is a non-responding sensor that does not generate any detection record. It is tagged as a shadowed round node (blue).

Example 1 reveals three major problems with passive sensors: (1) Even when the intruder is detected by multiple sensors, each sensor reports the intruder's appearance with a margin of error. Those detection records should be aggregated for a more accurate result; (2) Some false positive records are generated, such as $r_{2,1}$ and $r_{5,1}$. The system must filter them out; (3) The sensor, s_6, should send out a detection record but it fails to do so. It is a false negative.

Definition 2. (Valid Detection) Let $q_{k,j}$ be the position of intruder o_k in time t_j. A record $r_{i,j}$ is a valid detection if there exists an intruder o_k that $dist(cen(r_{i,j}), q_{k,j}) \leq rad(r_{i,j})$.

False positive and false negative records are caused by various reasons, such as the wind blowing and animal movements. Sensor reliability is a critical factor that impacts the quality of detection results. We introduce two measurements of the sensor's reliability, as defined below.

Definition 3. (Robustness) Let s be a sensor, the robustness $\varphi(s)$ denotes the proportion of valid detections in all the records generated by s.

Definition 4. (Sensitivity) Let s be a sensor, the sensitivity $\psi(s)$ denotes the probabilities that s sends out a

Notation	Explanation	Notation	Explanation
S	the sensor set	s_i, s_j, s_k	the sensors
O	the intruder set	o_i, o_j, o_k	the intruders
R_j	the record set in t_j	$r_{i,j}$	the detection record by sensor s_i at t_j
t_i, t_j	the time stamp	$q_{k,j}$	the position of o_k in t_j
$p_{i,j}$, $p_{k,j}$	the intruder appearance	$\tau(p_{i,j})$	the trustworthiness of $p_{i,j}$
L_i, L_k	the intruder trajectory	L_k^ω	the ω-recent trajectory
$\varphi(s_i)$	the robustness of s_i	$\psi(s_i)$	the sensitivity of s_i
$S(r_{i,j})$, $S(p_{k,j})$	the watching sensor set of $r_{i,j}$ / $p_{k,j}$	$S_r(r_{i,j})$, $S_r(p_{k,j})$	the responding sensor set of $r_{i,j}$ / $p_{k,j}$
$S_n(r_{i,j})$, $S_n(p_{k,j})$	the non-responding sensor set of $r_{i,j}$ / $p_{k,j}$	$E_j(L_k)$	the expectation of L_k
G_j	the watching network in time t_j	β	the decay factor

Figure 3: List of Notations

valid detection record when an intruder passes through s's watching area.

Knowledge of the sensor's robustness and sensitivity are important for filtering out false data. However, the two scores may change over time. In the beginning, the sensor's robustness and sensitivity are both high. As time elapses, sensors may be damaged by the harsh environment, or run out of battery power. Therefore, both scores will drop and they should be dynamically updated based on the detection results.

The intruder is an object entering the watching area. The system discovers the intruder's movement as an *intruder trajectory*, which is a sequence of *intruder appearances* in different timestamps.

Definition 5. (Intruder Trajectory) Let o_k be an intruder and t_j be a timestamp; the intruder appearance $p_{k,j}$ is a spatial coordinate estimated by the system to indicate o_k's position in t_j. The intruder trajectory L_k is a sequence of o_k's appearances, $L_k = \{p_{k,1}, p_{k,2}, \ldots, p_{k,n}\}$.

Since users are only interested in trajectories that are long enough, they may set a threshold δ on the trajectory size. In addition, the sensor data arrive continuously in a data stream format. The system cannot output the results after scanning the whole dataset. Users require intruder trajectories to be discovered in the data stream.

Now we formally define the problem of "mining lines in the sand".

Problem Statement: Let S be the set of sensors and $\mathbb{R}$ be the sensor data arriving by time, $\mathbb{R} = \{R_1, R_2, \ldots, R_j, \ldots\}$, where $R_j = \{r_{1,j}, r_{2,j}, \ldots, r_{m,j}\}$. The sensors' locations are fixed and their robustness and sensitivity scores are initialized. Given a length threshold δ, the task of "mining lines in the sand" is to discover the set of intruder trajectories $\mathbb{L}$=$\{L_1, L_2, \ldots, L_k\}$ in real time, where $size(L_k) \geq \delta$.

Note that the total number of intruders is not known in advance. *LiSM* is required to discover the trajectories of all the intruders entering the watching area. We will introduce the detailed techniques of *LiSM* in the following sections. Figure 3 lists the notations used throughout this paper.

3. THE WATCHING NETWORK

In Example 1, s_1, s_3 and s_4 all detect the appearance of intruder o_1. However, another nearby sensor, s_6, should detect the intruder but does not generate any record. Such non-responding sensor disagrees with its responding neighbors. Therefore, the first task of *LiSM* is to retrieve the hidden relationships of these sensors and intruders.

Definition 6. (Watching Sensors) Let S be the sensor set and $r_{i,j}$ be a detection record, the watching sensor set $S(r_{i,j}) = \{s | s \in S, dist(s, cen(r_{i,j})) < range(s) + rad(r_{i,j})\}$, where $dist(s, cen(r_{i,j}))$ denotes the distance between sensor s and the center of $r_{i,j}$, $range(s)$ is s's maximum sensing range.

Theoretically, if there is a real intruder appearing in the reported area of $r_{i,j}$, all the sensors of $S(r_{i,j})$ should send out detection records. However, only a subset of them send records to indicate the intruder's appearance around $r_{i,j}$. The set of watching sensors is partitioned into two parts of *responding sensors* and *non-responding sensors*.

Definition 7. (Responding Sensors) Let $r_{i,j}$ be a detection record, and $S(r_{i,j})$ be the watching sensor set of $r_{i,j}$, the responding sensor set $S_r(r_{i,j}) = \{s_k | s_k \in S(r_{i,j}), dist(cen(r_{k,j}), cen(r_{i,j})) \leq rad(r_{k,j}) + rad(r_{i,j})\}$, the non-responding sensor set $S_n(r_{i,j}) = S(r_{i,j}) - S_r(r_{i,j})$.

Based on Definitions 6 and 7, we can construct a watching network. This network contains nodes representing sensors and records. Two types of links are constructed in the network: positive links connect records to responding sensors and negative links connect records to non-responding sensors.

Example 2. Figure 4 shows a watching network constructed from the records in Example 1. For each record $r_{i,j}$, the system draws a circle with center at $cen(r_{i,j})$ and radius as $range(s) + rad(r_{i,j})$. The watching sensors $S(r_{i,j})$ are located inside this circle (We only draw a circle of $r_{4,1}$ in Figure 4 for simplicity). The system then connects records with positive links (solid lines) to responding sensors, and generates negative links (dashed lines) between records and non-responding sensors. Since sensor s_6 does not send any record, it has negative links to all the related records. Note that even though s_2 is a watching sensor of $r_{4,1}$ and s_2 sends out a detection record $r_{2,1}$, the distance between $cen(r_{4,1})$ and $cen(r_{2,1})$ is larger than $rad(r_{4,1})$+$rad(r_{2,1})$, thus the link between s_2 and $r_{4,1}$ is a negative link.

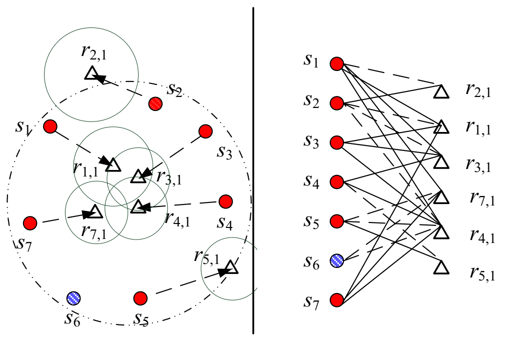


Figure 4: Example: The Watching Network

In the sensor data, many detection records are caused by the same intruder, *e.g.*, $r_{1,1}$, $r_{3,1}$ and $r_{4,1}$ are caused by intruder o_1. Such records are called *homologous records*.

Definition 8. (Homologous Records) Let $q_{k,j}$ be the position of intruder o_k in time t_j and R_j be the detection record set in t_j. The homologous record set $H_{k,j} = \{r | r_{i,j} \in R_j, dist(cen(r_{i,j}), q_{k,j}) \leq rad(r_{i,j})\}$.

If the intruder's position, $q_{k,j}$, is known in advance, the system can easily find the homologous records. However, the intruder's position is exactly required as the mining result. The system has to approximate the homologous records based on the following property.

Property 1. Let $H_{k,j}$ be a homologous record set in t_j, $r_{i,j}, r_{l,j} \in H_{k,j}$ be two records, and s_i, s_l be the sensors that send out those records. Then s_i is a responding sensor of $r_{l,j}$ and s_l is a responding sensor of $r_{i,j}$.

Proof: Let $q_{k,j}$ be the position of the corresponding intruder in $H_{k,j}$. According to Definition 8, $dist(cen(r_{i,j}), q_{k,j}) \leq rad(r_{i,j})$ and $dist(cen(r_{l,j}), q_{k,j}) \leq rad(r_{l,j})$.

Based on triangle inequality, $dist(cen(r_{i,j}),cen(r_{l,j})) \leq dist(cen(r_{i,j}),q_{k,j}) + dist(cen(r_{l,j}),q_{k,j}) \leq rad(r_{i,j}) + rad(r_{l,j})$.

By Definition 7, s_i is a responding sensor of $r_{l,j}$ and s_l is a responding sensor of $r_{i,j}$. ∎

The homologous record sets can be generated on the watching network. The system first picks a record as the seed to initialize a homologous record set, and retrieves all the responding sensors following the positive links. The records of those responding sensors are checked and added to the homologous record set.

Once the homologous record set is generated, we can estimate the position of an intruder appearance with Eq.1, where $\lambda_{i,j}$ is a normalized weight based on the radius of $r_{i,j}$. The records with with lower uncertainty (*i.e.*, smaller radius) have higher weights in determining the position of intruder appearance. Note that we adopt a linear model to compute $\lambda_{i,j}$ for general cases, the weight computation can be modified based on specific signal decay models of the sensors.

$$p_{k,j} = \sum_{r_{i,j} \in H_{k,j}} \lambda_{i,j} \cdot cen(r_{i,j})$$
$$\lambda_{i,j} = 1 - \frac{rad(r_{i,j})}{\sum\limits_{r_{l,j} \in H_{k,j}} rad(r_{l,j})} \quad (1)$$

Then the system retrieves the set of watching sensors for the newly computed intruder appearance and finds its responding and non-responding sensors.

Definition 9. Let S be the sensor set and $p_{k,j}$ be an intruder appearance, the watching sensor set $S(p_{k,j}) = \{s | s \in S, dist(s, p_{k,j}) < range(s)\}$.

Definition 10. Let $p_{k,j}$ be an intruder appearance, and $S(p_{k,j})$ is the watching sensor set of $p_{k,j}$, the responding sensor set $S_r(p_{k,j}) = \{s_i | s_i \in S(p_{k,j}),\ dist(p_{k,j},cen(r_{i,j})) \leq rad(r_{i,j})\}$, the non-responding sensor set $S_n(p_{k,j}) = S(p_{k,j})$-$S_r(p_{k,j})$.

The intruder appearances are added as new nodes to the watching network. Similarly, the positive and negative links are connected between the sensors and the appearances, as shown in Figure 5.

With the link information of the watching network, we can estimate the trustworthiness of each intruder appearance based on the sensor's robustness and sensitivity. For an appearance $p_{k,j}$, let $s_i \in S_r(p_{k,j})$ be a responding sensor, and $s_j \in S_n(p_{k,j})$ be non-responding sensor. If $p_{k,j}$ is a real appearance, then s_i reports a valid detection and s_j is a false negative. The probability of $p_{k,j}$ being a valid detection is calculated as Eq.2, where $\varphi(s_i)$ is the robustness of s_i and

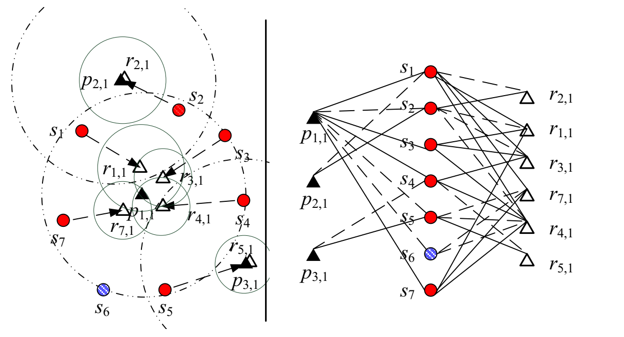


Figure 5: Example: The Watching Network with Intruder Appearances

$\psi(s_j)$ is the sensitivity of s_j .

$$\Pr(p_{k,j})^+ = \prod_{s_i \in S_r(p_{k,j})} \varphi(s_i) \cdot \prod_{s_j \in S_n(p_{k,j})} (1 - \psi(s_j)) \quad (2)$$

Similarly, the probability of $p_{k,j}$ being a false positive can be written as Eq.3.

$$\Pr(p_{k,j})^- = \prod_{s_i \in S_r(p_{k,j})} (1 - \varphi(s_i)) \cdot \prod_{s_j \in S_n(p_{k,j})} \psi(s_j) \quad (3)$$

The trustworthiness of intruder appearance, $\tau(p_{k,j})$, is then calculated as Eq.4.

$$\tau(p_{k,j}) = \log \frac{\Pr(p_{k,j})^+}{\Pr(p_{k,j})^-}$$
$$\propto \sum_{s_i \in S_r(p_{k,j})} \frac{\varphi(s_i)}{1 - \varphi(s_i)} + \sum_{s_j \in S_n(p_{k,j})} \frac{1 - \psi(s_j)}{\psi(s_j)} \quad (4)$$

Figure 6 lists the algorithm to detect the intruder appearances. The algorithm first scans each detection record and retrieves the responding and non-responding sensors (Lines 1 – 4). Then the system initializes the homologous record $H_{k,j}$ by randomly picking a seed record from the watching network (Lines 6 – 7). For each unvisited record $r_{i,j}$ in $H_{k,j}$, the algorithm retrieves $r_{i,j}$'s responding sensors and checks its record $r_{l,j}$. If $r_{l,j}$ does not belong to any existing homologous record sets and the distances from $r_{l,j}$ to all other records of $H_{k,j}$ is less than the sum of the radius, $r_{l,j}$ is then added to $H_{k,j}$ (Lines 8 – 14). Once $H_{k,j}$ is generated, the system calculates the intruder appearance $p_{k,j}$ and adds it to the network (Lines 15 – 18).

4. TRAJECTORY GENERATION

The watching network discovers the intruder appearances in each snapshot. It is an effective tool for "mining dots in the sand". However, a more critical task is "connecting the dots as lines". Since the intruders do not send out any identification information, the system has to distinguish them automatically.

After mining the intruder appearances in the first snapshot, *LiSM* initializes a set of candidate trajectories. Each candidate trajectory contains a discovered intruder appearance. In the following snapshots, the system matches the newly detected intruder appearances with the candidate trajectories. The appearances with the highest matching probabilities are added to the corresponding candidate trajectories.

```
Algorithm 1. The Intruder Appearance Detection
Input: The record set $R_j$ in time $t_j$, the sensor set $S$.
Output: The watching network $G_j$.

1.  initialize $G_j$;
2.  for each detection record $r_{i,j} \in R_j$
3.      add $r_{i,j}$ to $G_j$;
4.      compute $S_r(r_{i,j})$ and $S_n(r_{i,j})$, construct the links;
5.  repeat
6.      random select a record as the seed;
7.      initialize homologous record set $H_{k,j}$ by the seed;
8.      for each unvisited record $r_{i,j} \in H_{k,j}$
9.          tag $r_{i,j}$ as visited;
10.         for each responding sensor $s_l$ in $S_r(r_{i,j})$
11.           if $r_{l,j}$ does not belong to any record set
12.             for each record $r_{m,j} \in H_{k,j}$
13.               if $dist(cen(r_{m,j}), cen(r_{l,j})) \leq rad(r_{m,j}) + rad(r_{l,j})$
14.                 add $r_{l,j}$ to $H_{k,j}$;
15.     compute the intruder appearance $p_{k,j}$ from $H_{k,j}$;
16.     add $p_{k,j}$ to $G_j$;
17.     compute $S_r(p_{k,j})$ and $S_n(p_{k,j})$, construct the links;
18.     calculate $\tau(p_{k,j})$;
19. until all the records of $G_j$ are processed;
20. return $G_j$;
```

Figure 6: Algoirthm: The Intruder Appearance Detection

Let $p_{i,j}$ be an intruder appearance in time t_j, L_k be a candidate trajectory, the trustworthiness of $p_{i,j}$ belonging to L_k is computed as shown in Eq.5, where $\tau(p_{i,j})$ is the trustworthiness of $p_{i,j}$, and $P(p_{i,j}, L_k)$ is the matching probability of $p_{i,j}$ and L_k.

$$\tau(p_{i,j} \in L_k) = \tau(p_{i,j}) \cdot P(p_{i,j}, L_k) \tag{5}$$

The key issue is computing the matching probability between an intruder appearance and a candidate trajectory. To this end, we propose the *cone model*. This model stores the intruder's recent moving history and predicts the intruder's next move in a cone area. The detected intruder appearances are projected on to the area to compute the matching probability.

Definition 11. **(ω-recent Trajectory)** Let L_k be the trajectory of intruder o_k, t_j be the current timestamp and ω be a positive number, $\omega \leq size(L_k)$. The ω-recent trajectory L_k^ω is a subset of L_k, $L_k^\omega = \{p_{k,j-\omega}, p_{k,j-\omega+1}, \ldots, p_{k,j-1}\}$.

The ω-recent trajectory contains the ω-latest appearances of intruder o_k before time t_j. It is a short history of the intruder's movement. The system can calculate o_k's recent moving speed and direction based on L_k^ω. The mean and deviation of the intruder speed in period $[t_{j-\omega}, t_{j-1}]$ are calculated as shown in Eqs. 6 and 7.

$$\bar{v}_k = \frac{\sum_{i=j-\omega}^{j-2} dist(p_{k,i}, p_{k,i+1})}{(t_{j-1} - t_{j-\omega})} \tag{6}$$

$$\sigma(v_k) = \sqrt{\frac{\sum_{i=j-\omega}^{j-2} dist(p_{k,i}, p_{k,i+1})^2}{(t_{j-1} - t_{j-\omega})} - \bar{v}_k^2} \tag{7}$$

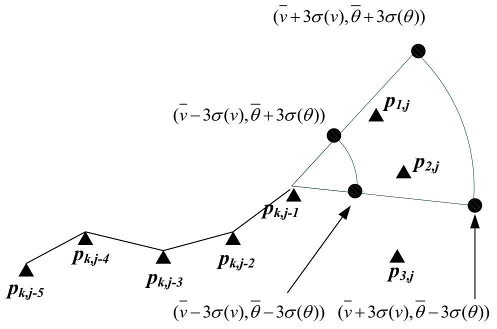
We use the function $direction(p_{k,i}, p_{k,i+1})$ to measure the angle between o_k's moving direction and the x-axis in time $[t_i, t_{i+1}]$. The mean and deviation of the moving direction are computed as shown in Eqs. 8 and 9.

$$\bar{\theta}_k = \frac{\sum_{i=j-\omega}^{j-2} direction(p_{k,i}, p_{k,i+1})}{(t_{j-1} - t_{j-\omega})} \tag{8}$$

$$\sigma(\theta_k) = \sqrt{\frac{\sum_{i=j-\omega}^{j-2} direction(p_{k,i}, p_{k,i+1})^2}{(t_{j-1} - t_{j-\omega})} - \bar{\theta}_k^2} \tag{9}$$

When intruders pass through the watching area, they are unlikely to change moving speed and direction dramatically. We make the assumption that the values of intruder speed and direction follow a normal distribution, and build a cone model to predict the area of o_k's appearance in t_j.

Example 3. Figure 7 shows the cone model for intruder o_k. Suppose ω is set to 5; the system retrieves o_k's latest five appearances as L_k^ω, and computes o_k's speed and direction. If those parameters follow a normal distribution, the probability is 99.7% that o_k' speed and direction of period $[t_{j-1}, t_j]$ are within three standard deviations of the mean values. The system calculates the four boundary points as shown in Figure 7. The area of o_k's next possible appearance is then generated as a partial cone with apex in $p_{k,j-1}$.



$(\bar{v}+3\sigma(v), \bar{\theta}+3\sigma(\theta))$
$(\bar{v}-3\sigma(v), \bar{\theta}+3\sigma(\theta))$
$p_{1,j}$
$p_{2,j}$
$p_{k,j-1}$
$p_{k,j-4}$
$p_{k,j-2}$
$p_{k,j-3}$
$p_{k,j-5}$
$p_{3,j}$
$(\bar{v}-3\sigma(v), \bar{\theta}-3\sigma(\theta))$
$(\bar{v}+3\sigma(v), \bar{\theta}-3\sigma(\theta))$

Figure 7: Example: The Cone Model

Let $p_{i,j}$ be an intruder appearance in the cone area, and $p_{k,j-1}$ be the latest intruder appearance of L_k^ω, if intruder o_k moves from $p_{k,j-1}$ to $p_{i,j}$, then o_k's speed and direction in $[t_{j-1}, t_j]$ are estimated as Eqs. 10 and 11.

$$\hat{v}_{k,j} = \frac{dist(p_{k,j-1}, p_{i,j})}{(t_j - t_{j-1})} \tag{10}$$

$$\hat{\theta}_{k,j} = \frac{direction(p_{k,j-1}, p_{i,j})}{(t_j - t_{j-1})} \tag{11}$$

By comparing $\hat{v}_{k,j}$ and $\hat{\theta}_{k,j}$, the system can estimate the matching probability between $p_{i,j}$ and L_k as Eq. 12.

$$P(p_{i,j}, L_k) = \frac{1}{\sqrt{2\pi}\sigma(v_k)} \exp\left(-\frac{(\hat{v}_{k,j} - \bar{v}_k)^2}{2\sigma(v_k)^2}\right) \cdot \frac{1}{\sqrt{2\pi}\sigma(\theta_k)} \exp\left(-\frac{(\hat{\theta}_{k,j} - \bar{v}_k)^2}{2\sigma(\theta_k)^2}\right) \tag{12}$$

Example 4. Suppose there are three intruder appearances detected in t_j, as shown in Figure 7. $p_{1,j}$ and $p_{2,j}$ locates in the cone area and $p_{3,j}$ is outside the area. Their trustworthiness scores are: $\tau(p_{1,j}) = 0.1$, $\tau(p_{2,j}) = 0.8$, $\tau(p_{3,j}) = 0.9$. Even $p_{3,j}$ has the highest trustworthiness, it is impossible to be an appearance of L_k. By considering the matching probability and trustworthiness of remaining two appearances, the system selects $p_{2,j}$ as the intruder's appearance in t_j.

Note that we make the assumption that the values of intruder speed and direction follow a normal distribution in this study. Based on our experiment results, this assumption works well. The cone model can be adopted to other distributions/models of the intruder movements.

If the trajectory L_k does not contain enough intruder appearances (*i.e.*, $size(L_k) \leq \omega$), the system will construct a cone model with default speed v_0 and $\sigma(v_0)$. The default parameters can be specified by the user, or calculated as the mean of all the other intruders' ω-recent trajectories. In such a case, the system also releases the constraint on movement direction (*i.e.*, the intruder may move in any direction). The matching probability is then written as Eq.13.

$$P(p_{i,j}, L_k) = \frac{1}{\sqrt{2\pi}\sigma(v_k)} \exp\left(-\frac{(\hat{v}_{k,j} - \bar{v}_k)^2}{2\sigma(v_k)^2}\right) \tag{13}$$

Figure 8 shows the detailed steps of trajectory tracking. For each candidate trajectory L_k, Algorithm 2 first checks the trajectory size. If the size is larger than ω, the system retrieves ω-recent trajectory L_k^{ω} and calculates the intruder's speed and direction. If the size of L_k is less than ω, the system uses the default parameters (Lines 2 – 5). Then the algorithm constructs the cone model. For each intruder appearance inside the cone area, the system calculates the matching probability. The one with the highest probability is tagged as "matched" and added in L_k (Lines 6 – 15). Finally the system initializes new candidate trajectories for the unmatched intruder appearances (Lines 16 – 18).

Algorithm 2. The Trajectory Tracking

Input: The candidate trajectory set $\mathbb{L}$, the watching network G_j in time t_j, the positive number ω.
Output: The updated trajectory set $\mathbb{L}$.

```
1.  for each trajectory $L_k \in \mathbb{L}$
2.      if size$(L_k) \geq \omega$
3.          retrieve the $\omega$-recent trajectory $L_k^{\omega}$;
4.          calculate the moving parameters of $L_k^{\omega}$;
5.      else $\bar{v} = v_0, \sigma(v) = \sigma(v)_0$;
6.      calculate the cone model of $L_k^{\omega}$;
7.      $\tau_{max} = 0$, $p_{k,j} \leftarrow \emptyset$;
8.      for unmatched appearance $p_{i,j}$ inside the cone area
9.          calculate $\tau(p_{i,j} \in L_k)$;
10.         if $\tau(p_{i,j} \in L_k) > \tau_{max}$
11.             $\tau_{max} = \tau(p_{i,j} \in L_k)$;
12.             $p_{k,j} \leftarrow p_{i,j}$
13.         if $p_{k,j} \neq \emptyset$
14.             add $p_{k,j}$ to $L_k$;
15.             tag $p_{k,j}$ as matched;
16. for unmatched intruder appearances $p_{i,j} \in G_j$
17.     initialize a trajectory $L_i$ by $p_{i,j}$;
18.     add $L_i$ to $\mathbb{L}$;
19. return $\mathbb{L}$;
```

Figure 8: Algorithm: The Trajectory Tracking

In Algorithm 2, the system initializes new trajectories based on unmatched intruder appearances in every snapshot. However, majority of them are "ghost trajectories". The ghost trajectories are generated by the untrustworthy appearances (false positives), such as $p_{2,1}$, $p_{3,1}$ in Figure 5. It is a burden for the system to maintain them in memory. When the time elapses, real trajectories grow longer with more subsequent appearances added in, but ghost trajectories are unlikely to get more appearances. Hence we can eventually prune them.

Definition 12. (Trajectory Expectation) Let L_k be a candidate trajectory and t_j be the current timestamp, the trajectory expectation $E_j(L_k)$ denotes the expectation that L_k is a qualified mining result in time t_j. $E_j(L_k)$ is defined as shown in Eq.14, where t_1 is the timestamp of the first intruder appearance in L_k, and β is a decay constant.

$$E_j(L_k) = \sum_{p_{k,i} \in L_k} \tau(p_{k,i}) - \beta(t_j - t_1) \tag{14}$$

In the end of every snapshot, the system checks the expectation of each candidate trajectory. If the expectation is less than zero, such a trajectory is unlikely to become a qualified result and should be removed from main memory. Meanwhile, if a trajectory's length is longer than the threshold δ, the system will report it to the user.

In many CPS applications, the sensors may be damaged by the environment or run out of battery power as time elapses; the system should also update the sensor's reliability scores.

Let L_k be a candidate trajectory, $L_k = \{p_{k,1}, p_{k,2}, \ldots, p_{k,n}\}$. If L_k is removed from the candidate set as a ghost trajectory, all the intruder appearances of L_k will be tagged as "ghost appearances". Let $p_{k,j}$ be such a ghost appearance. For all the responding sensors $s_i \in S_r(p_{k,j})$, s_i has reported a false positive, and its robustness should be reduced. $\varphi(s_i)$ is then updated as shown in Eq.15, where l_i is the number of false positives reported by s_i, and n_i is the total number of detection records generated by s_i.

$$\varphi(s_i) = 1 - \frac{l_i}{n_i} \tag{15}$$

Meanwhile, if L_k is output as a qualified mining result, all the intruder appearances of L_k are considered to be true. Let $p_{k,j}$ be a true appearance, for the non-responding sensor $s_j \in S_n(p_{k,j})$, s_j has made a false negative error, the sensitivity of s_i is then reduced as shown in Eq. 16, where f_i is the number of false negatives by s_i, m_i is the total number of intruders passed through s_i's watching area. Let l_i be the number of false positives by s_i, and n_i be the total number of detection records sent by s_i, $m_i = n_i - l_i + f_i$.

$$\psi(s_i) = 1 - \frac{f_i}{m_i} = 1 - \frac{f_i}{n_i - l_i + f_i} \tag{16}$$

5. PERFORMANCE EVALUATION

5.1 Experiment Setup

Datasets: To test the performance of *LiSM* in big and untrustworthy data, we generated four datasets based on the real military trajectories from the CBMANET project [5], in which an infantry battalion moves from Fort Dix to Lakehurst during a mission lasting 3 hours. The data generator retrieves 20 to 40 vehicle trajectories from CBMANET and simulates sensor monitoring fields along their routes with 200 to 10,000 deployed sensors. Each sensor scans the designated area every 10 seconds. If an intruder passes by, the sensor generates a detection record. The data generator randomly selects some sensors as false positive reporters, which may generate detection records without any local intruder. The set of false negative reporters is also generated, such sensors may not send detection record when an intruder passes by. The detailed features of those datasets are listed in Figure 9.

Baselines: The proposed *LiSM* algorithm (LM) is compared with two baselines: (1) The Karlman Filtering based method (KF); (2) TruAlarm method with nearest-neighboring tracking strategy (TA) [14].

Environments: The experiments are conducted on a PC with Intel 7500 Dual CPU 2.20G Hz and 3.00 GB RAM. The operating system is Windows 7 Enterprise. All the algorithms are implemented in Java on Eclipse 3.3.1 platform with JDK 1.5.0. The detailed parameter settings are listed in Figure 9.

Dataset	$\|O\|$	$\|S\|$	$\|R_i\|$	$\|\mathbb{R}\|$	fp%	fn%
D_1	10	225	27	$2.9*10^5$	10%	5%
D_2	20	2,500	57	$6.1*10^5$	20%	10%
D_3	30	3,600	121	$1.3*10^6$	40%	20%
D_4	40	10,000	326	$3.5*10^6$	50%	30%
$\|O\|$: intruder number, $\|S\|$: sensor number;						
$\|R_i\|$: the average size of detection record set in each snapshot;						
$\|\mathbb{R}\|$: the total size of the detection records;						
fp%: the false positive rate, **fn%**: the false negative rate;						
the size threshold δ: 4 – 16, default 12;						
the ω-recent trajectory ω: 3 – 8, default 6;						
the decay factor β: 0.05 – 0.2, default 0.05;						

Figure 9: Experiment Settings

5.2 Evaluations on Mining Efficiency

In the first experiment, we evaluate the efficiency of different algorithms with default parameters. The system processes LM, KF and TA on the four datasets and records their time costs. Figure 10(a) shows the results on the four datasets. Note that the y-axis is in logarithmic scale. In general, all three algorithms are efficient enough to process the data. LM achieves the best efficiency in all the cases, because the algorithm filters out low-expectation trajectory candidates in each snapshot and tracks the trajectories quickly with the cone model.

Then we study the factors that influence LM's efficiency. We set the decay factor β from 0.05 to 0.2 and record the algorithm's time cost on datasets D_1 to D_4 in Figure 10(b). With larger β, the system prunes more candidate trajectories and achieves better time efficiency. We also study the algorithm's running time with trajectory size threshold δ and the recent trajectory length ω. Both parameters do not influence the algorithm's efficiency, so we omit the results here.

5.3 Evaluations on Mining Effectiveness

To evaluate the quality of mining results, we retrieve the intruders' true trajectories as ground truth and compare against the mining results. There are two stage of "mining lines in the sand": (1) detecting the intruder appearances; (2) tracking their trajectories. In this experiment, we first

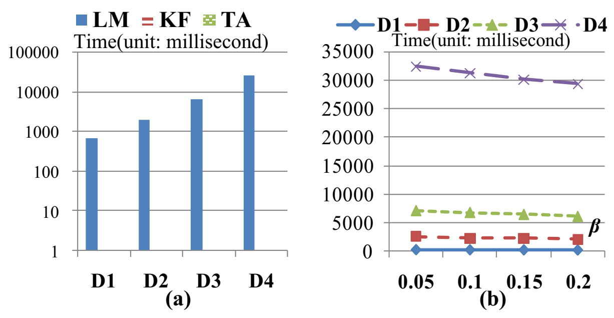


Figure 10: Efficiency: (a) time costs on different datasets and (b) influence of β.

compare the detected intruder appearances with the ground truth. If their distance is less than a reasonable error bound (20 meters), the detection is considered as a valid result. Then we check each generated trajectory L_k, if more than 90% of L_k's intruder appearances can be matched to a real trajectory in the ground truth, we consider L_k as a valid trajectory. Finally, we compute two measurements to evaluate the algorithms' effectiveness.

- Precision: The proportion of valid appearances/trajectories over the mining results. This represents the algorithm's selectivity for filtering out false positives.
- Recall: The proportion of valid appearances/trajectories over the ground truth. This criterion shows the algorithm's sensitivity for detecting the intruders.

The detection precision and recall of LM, KF and TA are shown in Figure 11. All the three methods can achieve a relative high recall of about 80%. However, the precision of KF and TA drops rapidly in D_3 and D_4, which have more untrustworthy data. The precision of KF is less than 20% in D_4, which is only one fourth of LM's precision. TA's precision is also lower than 50%. In contrast, LM filters out the false positive data and keeps the precision over 80%.

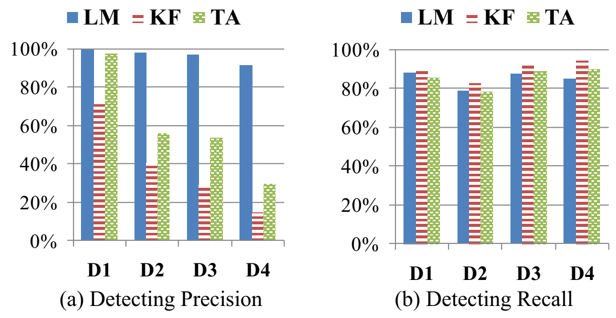


Figure 11: Effectiveness: Detecting (a) precision and (b) recall of intruder appearances on different datasets.

Then we check the tracking precision and recall of the three algorithms. The results are shown in Figure 12. TA's tracking performance is much worse than its detection effectiveness. The average precision is about 40% and the recall is less than 20%. This is caused by TA's tracking strategy: the nearest-neighboring tracking method always selects the nearest intruder appearance to add to the candidate trajectory. When there are multiple intruders whose trajectories intersect, the nearest-neighboring method is very likely to mix up their trajectories. KF's precision is also not high. This is due to the low precision of KF in the detection step. If the algorithm cannot detect the intruder appearances effectively in the first stage, the tracking results are inevitably influenced by the false positives. The precision and recall of LM are much higher; both of them are around 80%. These results indicate that LM is more suitable than TA and KF to process datasets with many untrustworthy reports.

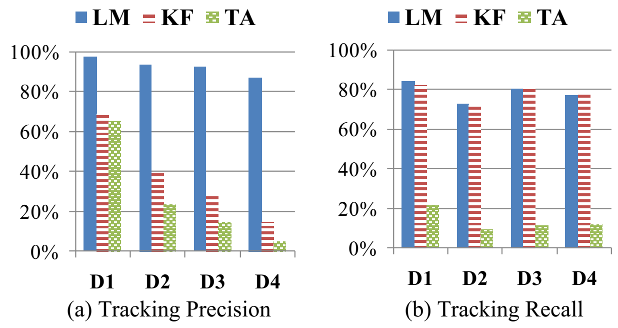


Figure 12: Effectiveness: Tracking (a) precision and (b) recall of intruder trajectories on different datasets.

In the next experiment, we investigate LM's precision and recall with different trajectory length threshold δ. The results are shown in Figures 13 and 14. With larger δ, fewer trajectories are reported. Hence the algorithm's precision increases, but the recall drops. Based on the experiment results, our suggestion is to select moderate δ (*e.g.*, 8 to 10) to make LM achieve the best performance.

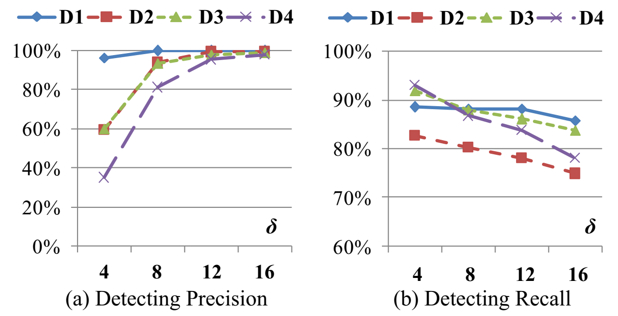


Figure 13: Effectiveness: Detecting (a) precision and (b) recall *w.r.t.* δ**.**

Finally, we study the influences of parameter ω and β. The results of LM's effectiveness are recorded in Figures 15 to 18. If the length of ω-recent trajectory is too short, LM may not be able to track the intruder with an accurate cone model. The decay factor β is used to filter the candidate trajectories; if it is set too large, the algorithm may prune some trustworthy candidates. The recall of LM is then reduced. Therefore, ω should be set as a reasonable large value (*e.g.*, 6 to 9) and β should be set relatively small (*e.g.*, 0.05).

6. RELATED WORK

The problem of mining trajectories from sensor data has received increasing attention in recent years, the related studies can be loosely classified into two categories.

Intruder Detection. Arora *et al.* propose the intrusion detection problem in wireless networks and design a detection model with acoustic and magnetic sensors [1]. Ozdemir *et al.* use the techniques of particle filtering to detect intruders [11]. Sheng and Hu propose the maximum likelihood-based estimation method [12] and Tang *et al.* propose the Tru-Alarm filtering method [14].

The main concern of these methods is to detect the intruders in a single snapshot, *i.e.*, without considering the temporal information of the intruders' movement. Some studies focus on saving sensors' energy and communication bandwidth, they try to provide an optimal sensor deployment plan. *LiSM* actually complements those technologies and improves the system's applicability.

Trajectory Tracking. Lin *et al.* propose a framework for the in-network intruder tracking [8]. Zhong *et al.* provide the techniques to track intruders with a sequence of alarming sensors [18]. Oh *et al.* propose the Markov Chain data association method for target tracking [10].

In these studies, the researchers assume that the targets' locations at each snapshot are already known. They focus on connecting the targets' locations at different snapshots to generate trajectories. However, as pointed out in [1], the intruder tracking results cannot be accurate based on many false intruder detections. To the best of our knowledge, *LiSM* is the first study to solve both detecting and tracking problems in an integrated framework.

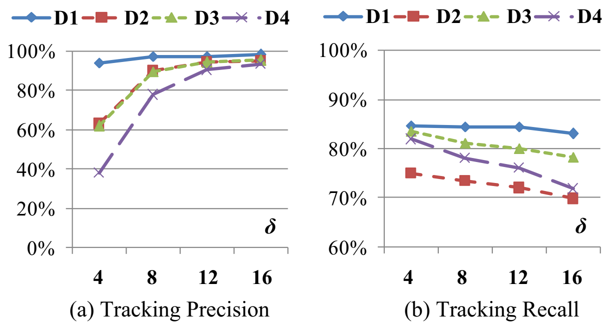


Figure 14: Effectiveness: Tracking (a) precision and (b) recall *w.r.t.* δ**.**

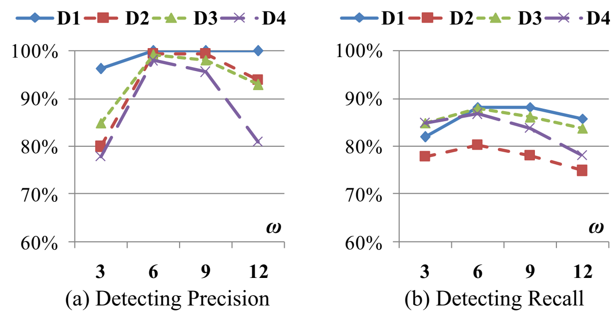


Figure 15: Effectiveness: Detecting (a) precision and (b) recall *w.r.t.* ω**.**

7. CONCLUSION AND FUTURE WORK

In this study we investigate the problem of mining trajectories in cyber-physical systems. We propose a novel

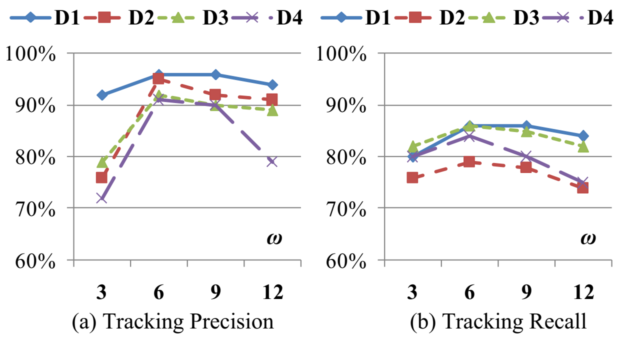


Figure 16: Effectiveness: Tracking (a) precision and (b) recall *w.r.t.* ω**.**

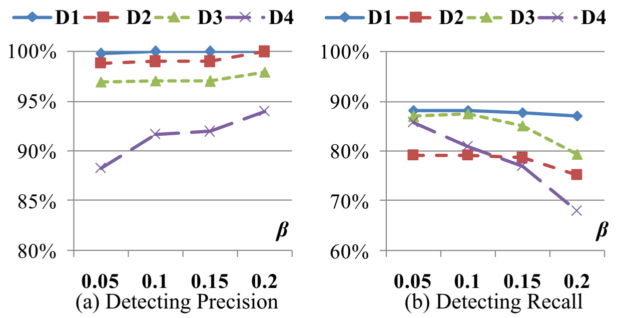


Figure 17: Effectiveness: Detecting (a) precision and (b) recall *w.r.t.* β**.**

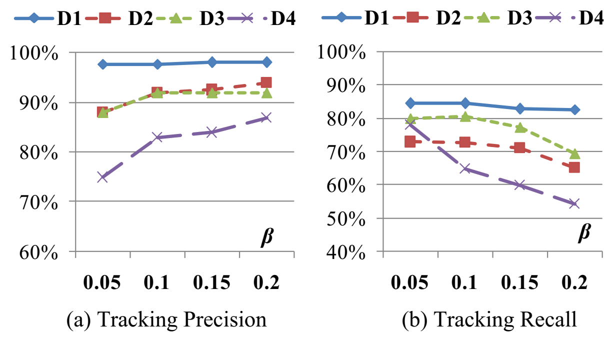


Figure 18: Effectiveness: Tracking (a) precision and (b) recall *w.r.t.* β**.**

method, *LiSM*, to discover intruder trajectories from untrustworthy sensor data. The watching network is designed to detect intruder appearances and the cone model is used to track their trajectories. We evaluate the proposed algorithms in extensive experiments on big datasets. *LiSM* outperforms the state-of-the-art methods on both detection and tracking tasks with higher precision and recall.

LiSM is proposed in 2D Euclidean environment. We are going to extend *LiSM* on more complicated scenarios such as the 3D environment, road networks and the indoor environment with obstacles. We are also interested in integrating more information, including the weather and local traffic, to improve the system performance.

8. ACKNOWLEDGEMENTS

The work was supported in part by U.S. National Science Foundation grants IIS-0905215, CNS-0931975, CCF-0905014, IIS-1017362, the U.S. Army Research Laboratory under Cooperative Agreement No.W911NF-09-2-0053 (NS-CTA) and W911NF-11-2-0086. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

9. REFERENCES

[1] A. Arora, P. Dutta, and S. Bapat. A line in the sand: a wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 46(5):605–634, 2004.

[2] T. N. S. Foundation. Cyber-physical systems. In *Program Announcements and Information*, 2008.

[3] M. Hewish. Reformatting fighter tactics. In *Jane's International Defense Review*, 2001.

[4] I. Hwang, H. Balakrishnan, K. Roy, and C. Tomlin. Multiple-target tracking and identity management in clutter, with application to aircraft tracking. In *Proceedings of the American Control Conference*, 2004.

[5] T. Krout. Cb manet scenario data distribution. In *BBN Tech. Report*, 2007.

[6] X. Li, R. Lu, X. Liang, X. Shen, J. Chen, and X. Lin. Smart community: an internet of things application. *IEEE Communications Magazine*, 49(11), 2011.

[7] Z. Li, J. Han, M. Ji, L. A. Tang, Y. Yu, B. Ding, J.-G. Lee, and R. Kays. Movemine: Mining moving object data for discovery of animal movement patterns. *ACM Transactions on Intelligent Systems and Technology*, 2(4), 2011.

[8] C. Lin, W. Peng, and Y. Tseng. Efficient in-network moving object tracking in wireless sensor network. *IEEE Transaction on Mobile Computing*, 5(8), 2006.

[9] C. Lo and W. Peng. Carweb: A traffic data collection platform. In *International Conference on Mobile Data Management*, 2008.

[10] S. Oh, S. Russell, and S. Sastry. Markov chain monte carlo data association for multi-target tracking. *IEEE Trans. Automat. Contr.*, 54(3), 2009.

[11] O. Ozdemir, R. Niu, and P. K. Varshney. Tracking in wireless sensor network using particle filtering: Physical layer considerations. In *IEEE Trans. on Signal Processing*, 2009.

[12] X. Sheng and Y. Hu. Maximum likelihood multiple source localization using acoustic energy measurements with wireless sensor networks. In *IEEE Trans. on Signal Processing*, 2005.

[13] R. Szewczyk, J. Polastre, and J. Mainwaring. Lessons from a sensor network expedition. In *European Workshop on Wireless Sensor Networks*, 2004.

[14] L. Tang, X. Yu, S. Kim, J. Han, C. Hung, and W. Peng. Tru-alarm: Trustworthiness analysis of sensor networks in cyber-physical systems. In *ICDM*, 2010.

[15] L. A. Tang, Q. Gu, X. Yu, J. Han, T. F. L. Porta, A. Leung, T. F. Abdelzaher, and L. M. Kaplan. Intrumine: Mining intruders in untrustworthy data of cyber-physical systems. In *SDM*, 2012.

[16] G. Tolle, J. Polastre, and R.Szewczyk. A macroscope in the redwoods. In *The ACM Conference on Embedded Networked Sensor Systems*, 2005.

[17] Y. Zheng and X. Zhou. *Computing with Spatial Trajectories*. Springer, 2011.

[18] Z. Zhong, T. Zhu, D. Wang, and T. He. Tracking with unreliable node sequences. In *INFOCOM*, 2009.