Knowledge-Based Systems 37 (2013) 346–356

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Trajectory mining from anonymous binary motion sensors in Smart Environment

Chengliang Wang [a,b,*], Debraj De [c], Wen-Zhan Song [c]

[a] *College of Computer Science, Chongqing University, Chongqing 400044, China*
[b] *School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta 30332, USA*
[c] *Department of Computer Science, Georgia State University, Atlanta 30303, USA*

ARTICLE INFO

Article history:
Received 19 April 2012
Received in revised form 6 August 2012
Accepted 18 August 2012
Available online 31 August 2012

Keywords:
Smart Environment
Trajectory mining
Wireless sensor networks
Behavior pattern
Motion sensor
Multiple pairs best trajectory algorithm

ABSTRACT

One of the key applications of Smart Environment (which is deployed with anonymous binary motion sensors [1,2]) is user activity behavior analysis. The necessary prerequisite to finding behavior knowledge of users is to mine trajectories from the massive amount of sensor data. However, it becomes more challenging when the Smart Environment has to use only non-invasive and binary sensing because of user privacy protection. Furthermore, the existing trajectory tracking algorithms mainly deal with tracking object either using sophisticated invasive and expensive sensors [3,4], or treating tracking as a Hidden Markov Model (HMM) which needs adequate training data set to obtain model's parameter [5]. So, it is imperative to propose a framework which can distinguish different trajectories only based on collected data from anonymous binary motion sensors. In this paper, we propose a framework – Mining Trajectory from Anonymous Binary Motion Sensor Data (*MiningTraMo*) – that can mine valuable and trust-worthy motion trajectories from the massive amount of sensor data. The proposed solution makes use of both temporal and spatial information to remove the system noise and ambiguity caused by motion crossover and overlapping. Meanwhile, *MiningTraMo* introduces Multiple Pairs Best Trajectory Problem (MPBT), which is inspired by the multiple pairs shortest path algorithm in [6], to search the most possible trajectory using walking speed variance when there are several trajectory candidates. The time complexity of the proposed algorithms are analyzed and the accuracy performance is evaluated by some designed experiments which not only have ground truth, but also are the typical situation for real application. The mining experiment using real history data from a smart workspace is also finished to find the user's behavior pattern.

1. Introduction

A Cyber-Physical System (CPS) is a system combining and establishing the interaction between the computational and physical resources. The Cyber-Physical System for Smart Environment (such as Smart Home, Smart Office, and Smart Building) (e.g. [7]) can: reason intelligently, act autonomously, and respond to the application needs in a context-aware manner [8]. One important function of Smart Environment is to obtain sequential activity patterns from the data collected from various sensors. Smart Environment can predict the inhabitants' upcoming actions or motion activities, using observed historical data [9]. Furthermore, the user motion trajectories hidden in these data can reveal the patterns of user behaviors, which could identify measures and quantify of social interactions, group behavior and organizational dynamics [10]. Thus, binary sensors (e.g., passive infrared motion sensors) have drawn considerable contributions ([11], etc.) for tracking based applications. The main advantages of using binary sensors are simplicity, non-invasive property, and minimal communication requirements.

As the precondition to these aforementioned applications, finding the motion trajectories generated by different users, is an important as well as difficult problem to be solved. This is because trajectory finding through user tracking is a hard problem due to requirements like: (i) ubiquitous applicability (therefore use of limited and cheaper sensing methods) and (ii) user privacies (use of non-invasive sensors). These requirements impose serious practical constraints on using expensive and invasive sensors [3]. Overall speaking, we can apply neither more sophisticated invasive and expensive sensors, nor can use some theoretical geometric models for sensing and tracking. Our designed system *MiningTraMo* (Mining Trajectory from Anonymous Binary Motion Sensor Data) does not rely on meticulous calibration and GPS localization, and does not need considerable data set with ground truth for model training as required by [5].

The main contributions of this paper are as follows:

[*] Corresponding author at: College of Computer Science, Chongqing University, Chongqing 400044, China. Tel.: +86 18983055830.

E-mail address: wangcl55@gmail.com (C. Wang).

0950-7051/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.knosys.2012.08.020

1. Designing *MiningTraMo* for trajectory mining from anonymous binary motion sensor data.
2. Providing algorithms for how to (a) find valid/meaningful trajectories and (b) remove the system noise and ambiguity, which is caused by user motion crossover and overlapping.
3. Conducting performance evaluation experiment.

The paper is organized as follows. The existing works in the literature is discussed in Section 2. The model definition and system design for Smart Environment is presented in Section 3. In Section 4, the trajectory mining framework and trajectory searching algorithm are discussed in detail, with theoretical proof and synthesized data. Then in Section 5 the performance our proposed framework are evaluated with the real data set collected from a Smart Workplace. Finally, in Section 6, we conclude this work and discuss the future research directions.

2. Related work

Wireless Sensor Networks (WSNs) have been widely applied in many social and scientific applications. The work in [3] uses sensor networks for multi-objective tracking. It studies the problem of deterministically associating a track, revealed by the sensor network, with the trajectory of a unique anonymous object (namely the *multiple object tracking and identification* (MOTI) problem). Most of the traditional methods of this type of path finding algorithms, such as those algorithms proposed in [12–14], are mainly analytical approaches. In [12], Razavian introduces Potential Field, Free Space Configuration and Vertex Graphs as the solution. It is suitable for Robotic Systems, but not for Smart Environment. In [13], *Real-time Accurate and Scalable System* (RASS) provides a system for transceiver-free user tracking with Radio-Frequency (RF) based technology. But this system requires high density RF-based sensors to be evenly deployed in the working environment. This is not always suitable for practical Smart Environment applications. The work in [14] proposed *Received Signal Strength* (RSS) measurements for target tracking. But using RSS is unreliable in different physical environments, thus limits its general applicability.

There are several proposed works for trajectory or path (of moving objects) searching (such as [15,16]). Some other works involve spatial–temporal data mining such as in [17,18]. However, most of these solutions cannot be applied into the problem of trajectory mining from anonymous binary motion sensor data set. The trajectory searching algorithms in [19,20] are based on the fact that *Personal Communication Systems* (PCSs) can store and update the location information of mobile users, who are served by the system. Another kind of path finding strategy (such as [21,1]) used GPS-based system. But GPS signals are not always effective in indoor environments. Moreover, the relevant algorithms, based on theory of Bayesian networks, Particle filters or Kalman filter, can hardly be directly applied to our problem.

Our group has done related work for years. In [22], the sensor platform, called TelosW, is invented and installed in a workplace environment in GSU department of computer science. One of this testbed's purpose is for the research of user behavior, which requires trajectory mining from the collected data sent by those deployed sensors. *FindingHumo*, introduced in [23], applies the technique of Hidden Markov Model (HMM) with Viterbi decoding to track the use's motion trajectory. However, great amount of delicate designed experiments have to be done to obtain the model's State Transition Probability A and Emission Probability Distribution C. So, we continue this tracking research and propose a more general trajectory mining work in this paper.

3. System model and design

In this section, we define the model for Smart Environment, equipped with motion sensors. Then we propose the way of finding a valid trajectory, that can help the system understand: the environment, functional relationships among it's rooms, and behavior pattern of users (such as in office building, medical building). *Transition Relative Distance Matrix* is proposed and discussed in details, to demonstrate why it is helpful to solve the problem of ambiguity. The working methods of the whole system are described step-by-step with the physical layout as shown in Fig. 1a. The deployed sensor node is TelosW (in [22] for detailed introduction) and sample position of sensor nodes are represented in Fig. 1b.

3.1. Basic definitions

Now we define the terminologies used in this work. Let $\mathbb{M} = \{m_1, m_2, \ldots, m_j, \ldots, m_N\}$ is the set of all the motion sensor nodes, which are installed across the Smart Environment space. N is the number of sensor nodes. The location of node m_j is used to represent the user's position when his movement is detected by m_j. $\mathbb{U} = \{u_1, u_2, \ldots, u_x\}$ is the set of the users moving in the Smart Environment where those motion sensors can detect their presence. $\mathbb{T} = \{t_0, t_1, \ldots\}$ is represented as a discrete and infinite sequence of time instants starting at time t_0. Users come into, walk around and leave this area during the period of time $\mathbb{T}$. The term $m_i \rightarrow m_j$ is denoted as the position transition for this user. So, there is a transition relative distance matrix, denoted as Ω. As Same as the work in [3], we also assume that the movements of $\mathbb{U}$ are always possible in both directions and there is only one single way to move between two positions. So, Ω should be a symmetric matrix. We discuss this more in Section 3.2. The position of a user u at a specific time instant t is given by the function $loc : \mathbb{T} \times \mathbb{U} \rightarrow \mathbb{M}$, which returns the corresponding sensor node m_j. Being different from the work of [3], $m_j = \phi$ can happen in our scenario. This is because we cannot assume that the user could always trigger at least one of the sensor nodes. The term $b_j(t)$ is the output of sensor node m_j at time t, $\forall m_j \in \mathbb{M}$.

$$b_j(t) = \begin{cases} 1 & \text{if } \exists u \in \mathbb{U} : loc_t(u) = m_j \\ 0 & \text{otherwise} \end{cases}$$

$B(t)$ is the vector of all outputs of the sensor node set M at time t. When the sensor network works for time $\mathbb{T}$, there is a matrix $\mathbb{B} = \{B(t_0), B(t_1), \ldots\}$ generated by this system. Then our goal is to find the motion trajectories of the users $\mathbb{U}$ from $\mathbb{B}$.

Following notations are used in our framework:

- $M_t = \left(m^t_{j_1}, \ldots, m^t_{j_r}, \ldots, m^t_{j_{N_t}}\right)$ is the set of motion sensor nodes that are triggered by users at time t, $N_t = Count(b_j(t) = 1)$, where $j \in \{1,N\}$, and $m^t_{j_r} \in \mathbb{M}$. Every M_i is an observation.
- $\mathbb{W} = \{M_{t_0}, \ldots, M_{t_T}\}$ is all the sets of sensors that have been triggered from time t_0 until time t_T, and $\mathbb{W} \subset \mathbb{B}$.
- N^t_{leave} is the number of the users signed *leave* at time t and N^t_{come} is the number of the users signed *come* at time t.
- N^t_{pre} is the number of the users who triggered sensor at time t, but are not *signed-leave* or *signed-come* user at the same time. So $N^t_{pre} = N_t - N^t_{leave} - N^t_{come}$.
- K_t is the number of moving users at time t. Then $K_t = K_{t-1} + N^t_{come} - N^t_{leave}$. It is noticed that N_t may be different from K_t, because one or more users may be walking without triggering any sensor at time t. In other words, $loc_t(u) = \phi$.

The purpose of our work is to search the most fitting trajectory set $\mathbb{P}$ which can explain why the user set U can trigger sensor set $\mathbb{M}$ from time period $\mathbb{T}$ to generate observation history $\mathbb{W}$.

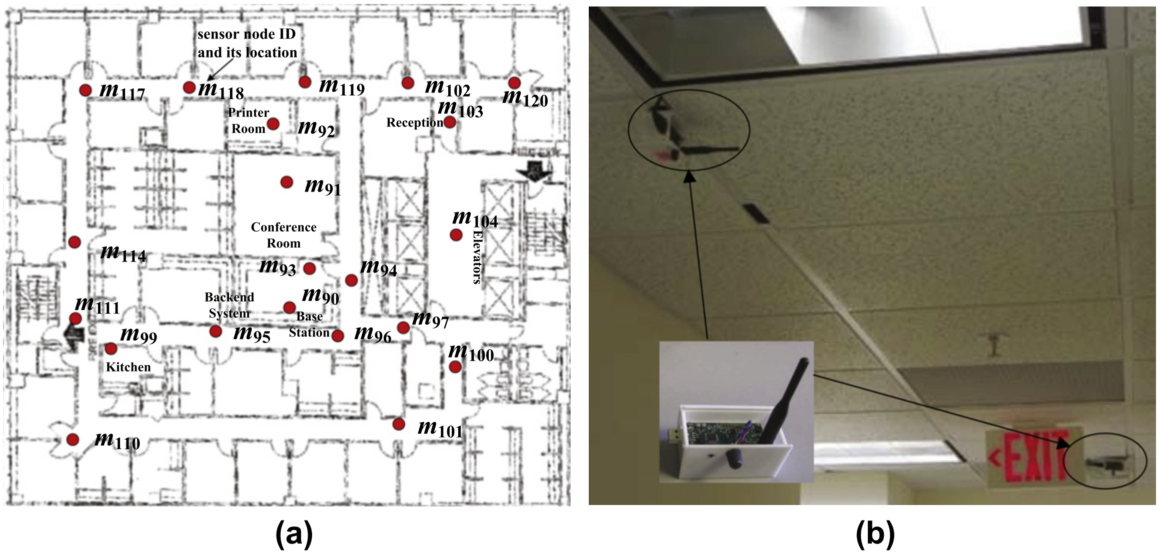


Fig. 1. A real Smart Environment with motion sensors installed. (a) Motion sensor network deployment in a Smart Workplace. (b) The smart environment example deployed with motion sensors.

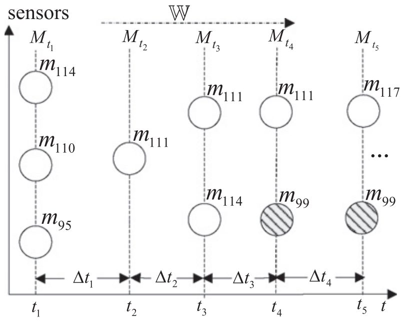


Fig. 2. Example of observation history W of three users u_1, u_2, u_3 in the period $[t_1 - t_5] = [0{:}00, 0{:}03, 0{:}05, 0{:}06, 0{:}08]$, $\Delta T = [\Delta t_1 - \Delta t_4] = [3,2,1,2]$ and there are multiple explanations for the real trajectories because of overlapping path.

3.2. Trajectory from binary sensor data

(1) Valid trajectory

A trajectory by a user u between time $t_i, t_j \in \mathbb{T}$ (with $t_i < t_j$) is defined as $P_{t_i,t_j,u} = [m^{t_i}, \ldots, m^{t_k}, \ldots, m^{t_j}]$, where $m^{t_k} = loc_{t_k}(u)$, $k \in [i,j]$. The number of sensor nodes, which appear in the trajectory, is trajectory length denoted as $C_{t_i,t_j,u}$. In real application, there are a lots of unexpected noise and interference. In order to mine more valuable knowledge from sensors triggering history data, *valid trajectory* should be defined first.

Definition 1. valid trajectory must begin with **bNode**, end with **eNode**, its length $C_{t_i,t_j,u}$ must be greater than *path-length-parameter* ξ. So, $P_{t_i,t_j,u} = [bNode_1^{t_i}, m_2^{t_{i+1}}, \ldots_{\ldots}, m^{t_k}, eNode_C^{t_j}]$, and $C > \xi$. The triggered sensor which cannot be included into any trajectory is defined as **noise-triggered sensor**, which could be caused by any kind of noise. And those trajectories, whose C is less than ξ, will not be treated as valid ones for mining work.

- *signed-come*: when a new user comes into this field, one of the marking sensors is assumed to be triggered, and it is denoted as **bNode**.
- *signed-leave*: when one user walks out of this field, one of the marking sensors is assumed to be triggered also, and it is denoted as **eNode**.

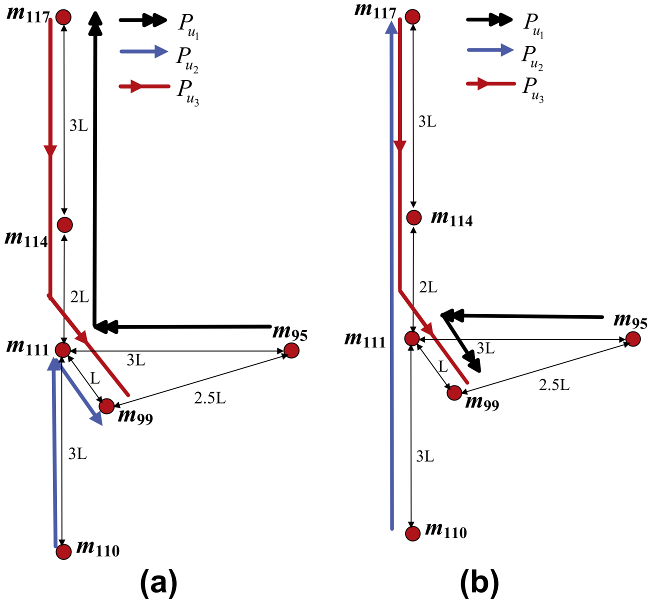
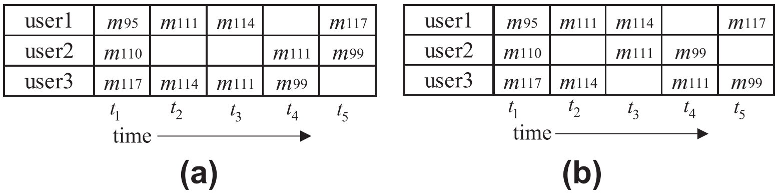


Fig. 3. Two possible trajectory options for the three users in the example shown in Fig. 2.

(2) Marking sensor

In order to discover these trajectories from binary sensor data, it is important and also possible to mark the beginning and the ending event for a user with **marking sensor** denoted with M_{mark}, which should be installed at all of possible entrance/exit. There are two kinds of marking sensors that can be installed in a Smart Environment. *The first one* is switch sensor, which will send On/Off signal when any entrance/exit's door is opened/closed. *The second one*, which is adopted in our research, is the motion sensor, whose triggered sequence can be used to mark the beginning (or ending) event. When these sensors are triggered, those adjacent triggered sensor (earlier or later) will generate a triggered sequence, which could indicate that a new user comes in or an existing user leaves out. Usually, these nodes are near to elevator, like 104, and located in some room, like 90, 91, 92, 93 and 99. When the *triggered sequence* is 104 → 103, or 93 → 94, that indicates a



user1	m_{95}	m_{111}	m_{114}		m_{117}
user2	m_{110}			m_{111}	m_{99}
user3	m_{117}	m_{114}	m_{111}	m_{99}	
	t_1	t_2	t_3	t_4	t_5

time
(a)

user1	m_{95}	m_{111}	m_{114}		m_{117}
user2	m_{110}		m_{111}	m_{99}	
user3	m_{117}	m_{114}		m_{111}	m_{99}
	t_1	t_2	t_3	t_4	t_5

time
(b)

Fig. 4. Detailed explanation for how the trajectories in Fig. 3a are generated.

		Sensors Index																				
		100	101	102	103	104	110	111	114	117	118	119	120	90	91	92	93	94	95	96	97	99
Sensors Index	100	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	101	2	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	102	0	0	0	1	0	0	0	0	0	0	2	2	0	0	0	0	4	0	0	0	0
	103	0	0	1	0	2	0	0	0	0	0	0	1.5	0	0	0	0	0	0	0	0	0
	104	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0
	110	0	6	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	111	0	0	0	0	0	2	0	1.5	0	0	0	0	0	0	0	0	0	0	0	0	1
	114	0	0	0	0	0	0	1.5	0	3	0	0	0	0	0	0	0	0	0	0	0	0
	117	0	0	0	0	0	0	0	3	0	2	0	0	0	0	0	0	0	0	0	0	0
	118	0	0	0	0	0	0	0	0	2	0	2	0	0	0	2	0	0	0	0	0	0
	119	0	0	2	0	0	0	0	0	0	2	0	0	0	2	1	6	6	0	0	0	0
	120	0	0	2	1.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0
	91	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	2	2	0	0	0	0
	92	0	0	0	0	0	0	0	0	0	2	1	0	0	0	0	0	0	0	0	0	0
	93	0	0	0	0	0	0	0	0	0	0	6	0	0	2	0	0	1	0	0	0	0
	94	0	0	4	0	0	0	0	0	0	0	6	0	1	2	0	1	0	0	1	0	0
	95	0	0	0	0	0	0	2.4	0	0	0	0	0	1	0	0	0	0	0	2	0	2
	96	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	2	0	2	0
	97	1	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0
	99	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	0	0	0

Fig. 5. Transition relative distance matrix of the system in Fig. 1a, 0 means there is no any chance to move between the two nodes (even itself), and the greater value indicates further reachable distance.

new coming event. When the triggered sequence is 97 → 104, or 95 → 99, that indicates a new leaving event.

3.3. Transition relative distance matrix

Taking the system in Fig. 1a as an illustrating example, if any user will trigger m_{111}, he must trigger one of $\{m_{114}, m_{99}, m_{95}, m_{110}\}$ first according to transition $m_{114} \rightarrow m_{111}$, $m_{99} \rightarrow m_{111}$, $m_{95} \rightarrow m_{111}$ or $m_{110} \rightarrow m_{111}$. If there is only one of $m_j \in \{m_{114}, m_{99}, m_{95}, m_{110}\}$ satisfying $b_j(t_{i-1}) = 1$, it is easy to infer that the trajectory from t_{i-1} to t_i for u is $m_j \rightarrow m_{111}$. However, if there are more than one user walking in the smart environment, and more than one sensor from $\{m_{114}, m_{99}, m_{95}, m_{110}\}$ are triggered at time t_{i-1} (as shown in Fig. 2), trajectory identifying becomes a *multiple object tracking and identification* (MOTI) problem defined by [3]. There are at least two kinds of trajectory combinations could generate observation $\mathbb{W}$ shown in Fig. 2, they are Fig. 3a and b. There are several sub-paths in Fig. 3a and b are overlapped by more than one user. Even though, there still are two possible detailed explanations for the proposition in Fig. 3a listed in Fig. 4a and b. So, $\mathbb{W}$ and the order of ΔT are not enough to find the most possible trajectory among several candidates. The other information, including the values of elements in ΔT, should be integrated also (see Fig. 5).

We know that different user u_i can walk with different speed s_i in a Smart Environment, but s_i is assumed slow-changing. In another word, user walking speed is treated as a constant or slow-changing variable. So, the distance between m_i and m_j, integrating with the time interval sequence $\Delta T = \{\Delta t_1, \Delta t_2, \ldots\}$, can be used to infer what the most possible trajectory combination is. Taking Fig. 4a as an example, if user 1 has spent Δt_1 to move a distance 3L, then Δt_2 should be very close to the value of $\frac{2}{3}\Delta t_1$. Thus the transition of $m_i \rightarrow m_j$ should not just reflect whether there is a reachable way from m_i to m_j (1 means Yes, 0 means No), how far the distance should also be given to reveal the influence of different walking speed, on time interval sequence ΔT. If the three trajectories are generated respectively according to Fig. 4a and b, then the following two relationships should be satisfied because users walking speed is assumed slow-changing during adjacent timeslot:

$$option1\begin{cases} S_{u_1} : \frac{3L}{\Delta t_1} \approx \frac{2L}{\Delta t_2} \approx \frac{3L}{\Delta t_3+\Delta t_4} (\checkmark) \\ S_{u_2} : \frac{3L}{\Delta t_1+\Delta t_2+\Delta t_3} \approx \frac{L}{\Delta t_4} (\checkmark) \\ S_{u_3} : \frac{3L}{\Delta t_1} \approx \frac{2L}{\Delta t_2} \approx \frac{L}{\Delta t_3} (\checkmark) \end{cases}$$

$$option2\begin{cases} S_{u_1} : \frac{3L}{\Delta t_1} \approx \frac{2L}{\Delta t_2} \approx \frac{3L}{\Delta t_3+\Delta t_4} (\checkmark) \\ S_{u_2} : \frac{3L}{\Delta t_1+\Delta t_2} \approx \frac{L}{\Delta t_3} (\times) \\ S_{u_3} : \frac{3L}{\Delta t_1} \approx \frac{2L}{\Delta t_2+\Delta t_3} \approx \frac{L}{\Delta t_4} (\times) \end{cases}$$

Considering the time interval $\Delta T = [3,2,1,2]$, it is easy to find that the relationship in *option 1* is satisfied instead of the one in *option 2*. However, the constraint relationship in Fig. 3a and b is the same one, this indicates that both of the two trajectories are reasonable based on current information. Because the distance between the nodes of trajectory $P_{u_1} = m_{95} \rightarrow m_{111}; m_{111} \rightarrow m_{114}; m_{114} \rightarrow m_{117}$; and $P_{u_2} = m_{110} \rightarrow m_{111}; m_{111} \rightarrow m_{99}$ in Fig. 3a is respectively as same as the distance in $P_{u_2} = m_{110} \rightarrow m_{111}; m_{111} \rightarrow m_{114}; m_{114} \rightarrow m_{117}$; and $P_{u_1} = m_{95} \rightarrow m_{111}; m_{111} \rightarrow m_{99}$ in Fig. 3b. So, the only way to distinguish them is to integrate some other information beyond this time period $[t_1 - t_5]$.

Then we use transition relative distance to define this distance relationship as follows:

Definition 2. transition relative distance $p_{m_i \leftrightarrow m_j}$ denotes that the relative distance of movement from m_i to m_j, both of which should be physically reachable without triggering any other nodes. $p_{m_i \leftrightarrow m_j} = 0$ means there is no reachability, and smaller value indicates nearer distance between them. Basically, we first set 0 to those physical unreachable relationships between sensors, and calculate transition relative distance using a standard distance value divided by the real distance measured from the deployment of sensors. The Table in Fig. 1a is the **transition relative distance matrix** of the system.

Actually, there are two kinds of knowledge in the matrix, the first one is physical *reachability* between sensors, which could be adopted to search the non-overlapping trajectory; the other is *distance relationship*, which, combined with ΔT, approximatively indicates the walking speed of user, and it could be introduced to refine the multiple overlapping trajectories.

4. Trajectory mining framework: *MiningTraMo*

The motion sensor installed in the Smart Environment will collect a huge amount of data. So, the trajectory searching task is suitable to be treated as a trajectory mining work from the binary motion data house. The corresponding mining framework *MiningTraMo* is represented as in Algorithm 1.

We find there are two situations making trajectory mining work challenging: crossover and overlapping, by which several possible explanations for collected data are caused. (Therefore, the key and time-consuming processes of *MiningTraMo* are overlapping detecting (in Section 4.1), crossover detecting (in Section 4.2) and searching and refining work (in Section 4.3), whose time complexity will be discussed.) We can treat a valid trajectory as a path connected with three kinds of sub-path: *disjoining*, *crossover* and *overlapping*.

Definition 3. When more than one users are walking in the Smart Environment in a same time period $\Delta\mathbb{T}$, and they walk across same sensor node m_j at different but very close time stage and then walk to different directions, then m_j is the crossover by these users.

Definition 4. In observation $\mathbb{W}_{\Delta\mathbb{T}}$, if there is a sub-path $P_{t_i,t_j,u}$, which is traveled by only one user during the time window $T_W = -t_j - t_i$, and there is not any *crossover* and the T_W is the maximum value to make this sub-path satisfy above condition, so $P_{t_i,t_j,u}$ is called *disjoining sub-path*.

Definition 5. When more than one user walking in the Smart Environment in a same time period $\Delta\mathbb{T}$, and their distinct trajectory cover at least one *same directed sub-path* after which there are at least two possible transitions, then there is overlapping in observation $\mathbb{W}_{\Delta\mathbb{T}}$, this *same directed sub-path* is called *overlapping sub-path*. When the length of this *overlapping sub-path* is equal to 1, *overlapping* becomes *crossover*.

There are two important indexes introduced in our mining framework: first, the *average-speed* is denoted with u_s^{id} where *id* is the identification of one sub-path $[m^{t_i}, \ldots, m^{t_k}, \ldots, m^{t_j}]$, and we can get its corresponding speed list $[s_{m^{t_i} \to m^{t_{i+1}}}, \ldots, s_{m^{t_{j-1}} \to m^{t_j}}]$ for every connected node-pair, so: $u_s^{id} = \frac{\sum_{k=i}^{j-1} s_{m^{t_k} \to m^{t_{k+1}}}}{j-i}$.

Second, the variance-speed is denoted with σ_s^{id} indicating the variance of the speed list $[s_{m^{t_i} \to m^{t_{i+1}}}, \ldots, s_{m^{t_{j-1}} \to m^{t_j}}]$.

Algorithm 1. Trajectories mining algorithm: *MiningTraMo*

```
Input: $\mathbb{W} = \{M_1, \ldots, M_T\}$, all the sets of observation from the
    beginning until timeslot $T$, we want to search all possible
    trajectories hidden in these observation sets; $\mathbb{T}$ time
    sequence; Transition Relative Distance Matrix $\omega$; marking
    sensors $M_{mark}$.
Output: Trajectory set $\Upsilon = (r_1, r_2, \ldots, r_D)$;
Constant: time-constraint-parameter $\tau$ and path-length-
    parameter $\xi$;
1: Signed come set $\Psi$ and leave set $Z$ from
    Signingpro$(\mathbb{W}, M_{mark})$
2: While $\Psi$ and $Z$ is not $\phi$ do
  2.1: $\theta^{N_{leave}}$ = extract the earliest leave(s) subset from $Z$; get
  subset $\mathbb{W}_{tc \to tl}$, $\Psi_{tc \to tl}$ and $\mathbb{T}_{tc \to tl}$ according to the time of the
  earliest leave(s) $t_l$ and the earliest come(s) $t_c$;
  2.2: (overlapping set $\Gamma$, disjoining set
       $\Theta_\Gamma$) = OverlapDetecting$(T_{tc \to tl}, \mathbb{W}_{tc \to tl}, \Omega, \tau)$;
       $\widetilde{\mathbb{W}}_{tc \to tl} = \textbf{label}\mathbb{W}_{tc \to tl}$ with $\Gamma$ and $\Theta_\Gamma$;
  2.3: (crossover set $\Phi$, disjoining set $\Theta_\Phi$)=CrossoverDetect-
       ing$(\mathbb{T}_{tc \to tl}, \mathbb{W}_{tc \to tl}, \Omega, \tau)$;
       $\widetilde{\mathbb{W}}_{tc \to tl} = \textbf{label}\mathbb{W}_{tc \to tl}$ with $\Phi$ and $\Theta_\Phi$;
  2.4: $\Upsilon_t^{N_{leave}^t}$ =SearchTraj$(\mathbb{T}_{tc \to tl}, \widetilde{\mathbb{W}}_{tc \to tl}, \Psi_{tc \to tl}, \theta^{N_{leave}}, \Omega, \xi)$;
  2.5: delete the head of every trajectory $\Upsilon_t^{N_{leave}^t}$ from $\Psi$;
  2.6: shrink $\mathbb{W}$ and $\mathbb{T}$ until next valid come; add $\Upsilon^{N_{leave}}$ into
       $\Upsilon$; return $\Upsilon$
```

4.1. Labeling: Overlapping detecting algorithm

Because overlapping indicates that there are at least two trajectories sharing the same sub-path at different moment during a same period, it is an obvious intuition that: more trajectory candidates is more time-consuming to find the best one. Such as the sub-path $m_{111} \to m_{99}$ are overlapped by two users, so there are more trajectories needed be evaluated to find the most possible ones.

Theorem 4.1. *Based on the presupposition that the sensor will be triggered once only when user passes by it, overlapping exists in observation* $\mathbb{W}_{\Delta\mathbb{T}}$ *if and only if:*

$$\begin{aligned}
&\exists m_{i1}, m_{j1}, m_{i2}, m_{j2}, m_{k1}, m_{k2} \in \mathbb{W}_{\Delta\mathbb{T}}; t_{m_{i1}} < t_{m_{j1}}, t_{m_{i2}} < t_{m_{j2}};\\
&(a) m_{i1} = m_{i2} = m_i \wedge m_{j1} = m_{j2} = m_j \wedge p_{m_i \leftrightarrow m_j} > 0; m_i, m_j \in \mathbb{M}\\
&(b) m_{k1} \neq m_{k2} \wedge p_{m_j \leftrightarrow m_{k1}} > 0 \wedge p_{m_j \leftrightarrow m_{k2}} > 0 \wedge (t_{m_{j1}}, t_{m_{j2}})\\
&\qquad < t_{m_{k1}} \wedge (t_{m_{j1}}, t_{m_{j2}}) < t_{m_{k2}}
\end{aligned} \tag{1}$$

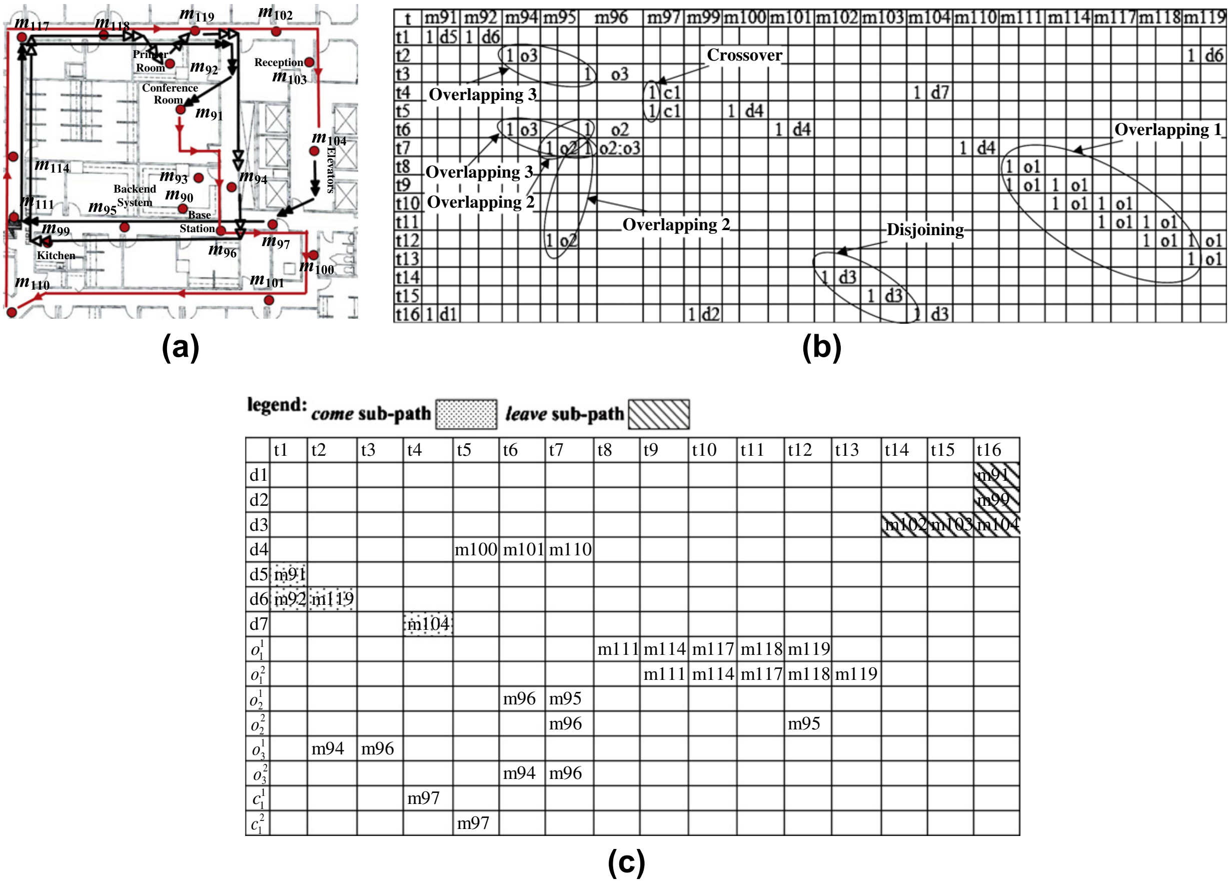
where t_{m_i} *is the time when sensor* m_i *was triggered by motion.*

In another plain word, there is a sensor node m_i in $\mathbb{W}_{\Delta\mathbb{T}}$ hold the following rules:

1. Both m_i and its reachable node m_j should join-appear more than one time together in $\mathbb{W}_{\Delta\mathbb{T}}$ and the order of their join-appearance should be same.
2. There is not any other sensor nodes m_k to which both m_i and m_j could transit appears in $\mathbb{W}_{\Delta\mathbb{T}}$ between when m_i and m_j was triggered.

Proof 1. Given (1) Formula in Eqs. (1) and (2) there is no any overlapping in observation $W_{\Delta T}$. According to Definition 1 and (1), we know there must be more than one user walking and sub-path $m_i \to m_j$ has been visited at least twice at different moment $t_{m_{i1}} \to t_{m_{j1}}$ and $t_{m_{i2}} \to t_{m_{j2}}$. So it is obvious that the result caused by different user's overlapping, which contradicts assumption (2).

Let us prove now that if there is overlapping in observation $\mathbb{W}_{\Delta\mathbb{T}}$, then Formula in Eq. (1) is true. According to Definition 5, there is at least one directed sub-path, like $m_i \to m_j$, has been visited by different users u_1 and u_2 for example, at different time, such as u_1 triggered m_i at $t_{m_{i1}}$, m_j at $t_{m_{j1}}$ and u_2 triggered m_i at $t_{m_{i2}}$ and m_j at $t_{m_{j2}}$. So $t_{m_{i1}} < t_{m_{j1}}$ and $t_{m_{i2}} < t_{m_{j2}}$ should hold be true together and there must be a possible transition from m_i to m_j also, indicating $p_{m_i \leftrightarrow m_j} > 0$. And, if there are two extra transition nodes, like m_{k1} and m_{k2}, can reach with m_j, and has been respectively triggered at the time $t_{m_{k1}}$, $t_{m_{k2}}$ when are both later than $t_{m_{j1}}, t_{m_{j2}}$. So



t	m91	m92	m94	m95	m96	m97	m99	m100	m101	m102	m103	m104	m110	m111	m114	m117	m118	m119
t1	1 d5	1 d6																
t2			1 o3															1 d6
t3					1 o3													
t4						1 c1						1 d7						
t5						1 c1		1 d4										
t6			1 o3		1 o2				1 d4									
t7				1 o2	1 o2:o3								1 d4					
t8														1 o1				
t9														1 o1	1 o1			
t10															1 o1	1 o1		
t11																1 o1	1 o1	
t12				1 o2													1 o1	1 o1
t13																		1 o1
t14										1 d3								
t15											1 d3							
t16	1 d1						1 d2					1 d3						

legend: *come* sub-path ░ *leave* sub-path ▨

	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12	t13	t14	t15	t16
d1																▨ m91
d2																▨ m99
d3														▨ m102	▨ m103	▨ m104
d4					m100	m101	m110									
d5	░ m91															
d6	░ m92	░ m119														
d7				░ m104												
o_1^1								m111	m114	m117	m118	m119				
o_1^2									m111	m114	m117	m118	m119			
o_2^1						m96	m95									
o_2^2							m96					m95				
o_3^1		m94	m96													
o_3^2						m94	m96									
c_1^1				m97												
c_1^2					m97											

Fig. 6. Illustration of sub-path classifications for a real example. (a) Three real trajectories in that smart environment. (b) The result after overlapping and crossover detecting, all of collected motion signals are labeled with Algorithms 2 and 3. (c) The labeled signals are classified into different sub-paths with their own *id*, d_i: the *i*th disjoining sub-path; o_i^j: *i*th overlapping sub-path visited for *j* times; c_i^j: *i*th crossover sub-path visited for *j* times.

it is difficult to distinguish the trajectory of u_1 is $P_{u1} = m_{i1} \to m_{j1} \to m_{k1} \to \ldots$ or $P_{u1} = m_{i1} \to m_{j1} \to m_{k2} \to \ldots m_i \to m_j$ is the *overlapping sub-path*. So the equivalence in Eq. (1) is proved. □

This sufficient and necessary condition on overlapping illustrates that overlapping is not only spatial relationship, like sub-path $m_i \to m_j$ shared by different users, but temporal relationship also. If $t_{m_{i1}} < t_{m_{k1}} < t_{m_{i2}}$, it is obvious that there is no chance for $u2$ to move from m_{j2} to m_{k1} and the trajectory for u_2 should be $P_{u2} = m_{i2} \to m_{j2} \to m_{k2} \to \ldots$ Based on Theorem 4.1, we have Algorithm 2 to detect whether overlapping is in observation $\mathbb{W}_{\Delta\mathbb{T}}$.

Algorithm 2 is illustrated in Fig. 6b, let $|N|$be the sensors number, $|T|$be the average number of the triggered sensors at every time index, and $|E|$be the edge number of Ω. Therefore, it takes $O(|T||N|)$ to search candidate nodes appeared in $\Delta\mathbb{W}$ more than once, and takes $O(|E||M|)$ to find all of pairs which are reachable from each other based on physical reachability of Ω for a candidate node, where $|M|$is the number of found candidate overlapping nodes. Thus, the time complexity of Algorithm 2 is $O(|T||N| + |E||M|^2)$.

Algorithm 2. Overlapping Detecting Algorithm

```
Input: time subsequence $\Delta\mathbb{T}$; sub-observation set $\mathbb{W}_{\Delta\mathbb{T}}$;
    Transition Relative Distance Matrix $\Omega$; time-constraint-
    parameter $\tau$.
Output: Overlapping sub-path set $\Gamma$; disjoining set $\Theta$.
1: $O$ = find all of nodes which appear in $\mathbb{W}_{\Delta\mathbb{T}}$ more than once;
    $\overline{O}$ = all the left nodes;
2: $(\Gamma, t_\Gamma)$ = In $O$, find all of pairs which are reachable from
    each other based on physical reachability of $\Omega$, where $t_\Gamma$ is
    every corresponding triggered time; add the left $O$ into $\overline{O}$;
3: $\Gamma$ = match the time order from those repeat-appearing
    pairs in $\Gamma$, keep those consistent pairs; add the unmatched
    $\Gamma$ into $\overline{O}$;
4: for every pair $m_i \to m_j$ in $\Gamma$ do
      find reachable nodelist of $m_i$ as $M_{ks}$ and corresponding
    triggered time set $T_{ks}$;
      if the time constraint is not satisfied for these triggered
    time in $T_{ks}$ then
        remove pair $m_i \to m_j$ from $\Gamma$ into $\overline{O}$;
5: for all nodes in $\overline{O}$ do
      $\theta_k$ = trace back from the latest nodes in $\overline{O}$ based on
    physical reachability of $\Gamma$ until unreachable or beyond the
    time-constraint-parameter $\tau$;
      add $\theta_k$ into $\Theta$;
  return $\Gamma$ and $\Theta$
```

4.2. Labeling: Crossover detecting algorithm

Comparing Definitions 3 and 5, we find that if the length of overlapping sub-path is equal to 1, then overlapping becomes crossover. Thus the framework of Crossover Detecting Algorithm, shown in Algorithm 3, resembles the main flow of Algorithm 2 and the complexity of this algorithm is $O(|T||N| + |E||M|^2)$ also, where $|M|$is number of the found candidate crossover nodes.

Algorithm 3. Crossover Detecting Algorithm

```
Input: time subsequence $\Delta\mathbb{T}$; sub-observation set $\mathbb{W}_{\Delta\mathbb{T}}$;
    Transition Relative Distance Matrix $\Omega$; time-constraint-
    parameter $\tau$.
Output: Crossover sub-path set $\Phi$; disjoining set $\Theta$.
1: $O$ = find all of nodes which appear in $\mathbb{W}_{\Delta\mathbb{T}}$ more than once;
    $\overline{O}$ =all the left nodes;
2: $(\Phi, t_\Phi)$ = In $O$, find those re-appearing nodes, whose
    triggered time is closer than time-constraint-parameter $\tau$;
    add the left $O$ into $\overline{O}$;
3: for every node $m_i$ in $\Phi$ do
  3.1: find reachable successive nodelist of $m_i$ in $\mathbb{W}_{\Delta\mathbb{T}}$ as $M_{ks}$
       and corresponding triggered time set $T_{ks}$;
  3.2: if $M_{ks} \rightarrow Count \leqslant 1$ or time constraint is not satisfied for
       these triggered time in $T_{ks}$ then  remove pair $m_i$ from
       $\Phi$ into $\overline{O}$;
4: for all nodes in $\overline{O}$ do
$\theta_k$ = trace back from the latest nodes in $\overline{O}$ based on physical
    reachability of $\Omega$ until unreachable or beyond the time-
    constraint $\tau$;
add $\theta_k$ into $\Theta$;
return $\Phi$ and $\Theta$
```

4.3. Trajectory searching algorithm

(1) Mainframe of algorithm

After the processes of 2.2 and 2.3 shown in Algorithm 1, all the nodes in observation $\mathbb{W}_{tc \rightarrow tl}$ have been labeled with nodes type (disjoining, crossover, or overlapping), and sub-path index (sub-path classification). So, the left work is to connect these sub-paths into trajectories with highest possibility. The pseudocode of the searching algorithm is given in Algorithm 4.

Algorithm 4. Trajectory Searching Algorithm

```
Input: time subsequence $\Delta\mathbb{T} = [t_1, t_2, t_3, \ldots, t_k]$; labeled sub-
    observation set $\widetilde{\mathbb{W}_{\Delta\mathbb{T}}}$; come subset $\Psi$; leave subset $\theta^{N_{leave}}$;
    Transition Relative Distance Matrix $\Omega$; path-length-
    parameter $\xi$;
Output: Trajectory subset $\Upsilon^{N_{leave}}$.
1: find parent–child relationships for every sub-path in $\widetilde{\mathbb{W}_{\Delta\mathbb{T}}}$
    based on physical reachability of $\Omega$ among all of sub-path
    set and time subsequence $\Delta\mathbb{T}$;
2: build connecting diagram $\mathbb{Y}$ with parent–child
    relationships;
3: delete the relationship in $\mathbb{Y}$ which is against the constraint
    (2);
4: delete the relationship in $\mathbb{Y}$ which is against the constraint
    (3);
5: repeat → find unambiguous trajectory $\Upsilon^a$ in $Y$ and delete
    these nodes of $\Upsilon^a$ from $\mathbb{Y}$, and insert $\Upsilon^a$ whose length
    $(\Upsilon^a) > \xi$ into $\Upsilon^{N_{leave}}$; → until there is none.
6: find optimal trajectory option using algorithm MPBT with
    parameters $Y,\ \Omega,\ \Delta\mathbb{T},\ \Psi,\ \theta^{N_{leave}}$ and the constraint (1), and
    insert these found trajectories into $\Upsilon^{N_{leave}}$;
  return $\Upsilon^{N_{leave}}$
```

There are several constraints to be introduced into for the sake of finding more meaningful and reasonable trajectories:

1. The connecting diagram is composed with several trajectories, every one of which starts with the *come* sub-path, follows with other sub-paths one by one, and ends with *leave* sub-path. All of those sub-path has following parameters: average-speed u_s^{id}, variance-speed σ_s^{id}, the time when the user walked into this sub-path t_{head}^{id}, and the time he walked away t_{tail}^{id}.
2. The time t_{tail}^{i} when the user walked away from sub-path i should be earlier than the time t_{head}^{i+1} when that user walked into the successive sub-path $i + 1$ of ith one.
3. Every sub-path should be connected with one and only one other sub-path. More specifically, every sub-path, except *come* one, should have one and one only parent sub-path; every sub-path, except *leave* one, should have one and one only child sub-path.

(2) Multiple pairs best trajectory problem (MPBT)

From Fig. 7d, the first trajectory starts with d_5 sub-path and the following successive ones are o_3^1 and c_1^1. There are two candidates o_2^1 and d_4 as the successor of c_1^1 and o_2^1, respectively have two possible successive sub-paths also, they are o_1^1 and o_1^2, which both has d_1 and d_3 as its potential tail sub-path. Hence, we notice that there are still $2 \times 2 \times 2 = 8$ possible different trajectories. So does the second trajectory which starts with d_7 and follows with c_1^2. Inspired by the multiple pairs shortest path algorithm in [6], we propose Algorithm MPBT that focus not on finding shortest path, but on searching the most stable speed list hidden behind these candidate trajectories. The pseudocode of **MPBT** is presented in Algorithm 5 and illustrated with the example of Fig. 6a–c.

Algorithms 4 and 5 which is respectively illustrated in Figs. 7 and 8. The whole process could be treated as connecting the nodes in graphic G based on physical reachability Ω. Let $|V|$ be the node number of G and $|E|$be the edge number of Ω. Therefore, it takes $O\left(\frac{|E||V|^2}{2}\right)$ to finish the connecting work and much less than $O(|V|)$ to delete the relationship against the constraints. The MPBT algorithm takes no more than $O(|M||V|)$ to compute optional trajectories' relative speed and their average variances. Thus, the time complexity of Algorithms 4 and 5 is $O\left(\frac{|E||V|^2}{2} + |M||V|\right)$.

Algorithm 5. MPBT Algorithm

```
Input: time subsequence $\Delta\mathbb{T} = [t_1, t_2, \ldots, t_k]$, connecting
    diagram $\mathbb{Y}$, come subset $\Psi$, leave subset $\theta^{N_{leave}}$, Transition
    Relative Distance Matrix $\Omega$.
Output: Trajectory subset $\Upsilon^{N_{leave}}$
Constant: path-length-parameter $\xi$
1: $\mathbb{H}$ = list all possible options for different trajectory
    combinations through path-expanding for connecting
    diagram $\mathbb{Y}$ with $\Psi$ and $\theta^{N_{leave}}$;
2: calculate relative speed $u_s$ for every sub-path of all
    trajectories in $\mathbb{H}$ based on distance relationship of $\Omega$:
    transition relative distance matrix, and $\Delta\mathbb{T}$;
3: calculate speed variance $\sigma_s$ of every trajectory in $\mathbb{H}$;
4: calculate the average value of the speed variance $\bar{\sigma}_s$ in the
    same option;
5: find the minimum $\bar{\sigma}_s$ as the optimal trajectory
    combination, and insert these found trajectories, whose
    $length(\Upsilon^a) > \xi$ into $\Upsilon^{N_{leave}}$;
  return $\Upsilon^{N_{leave}}$;
```

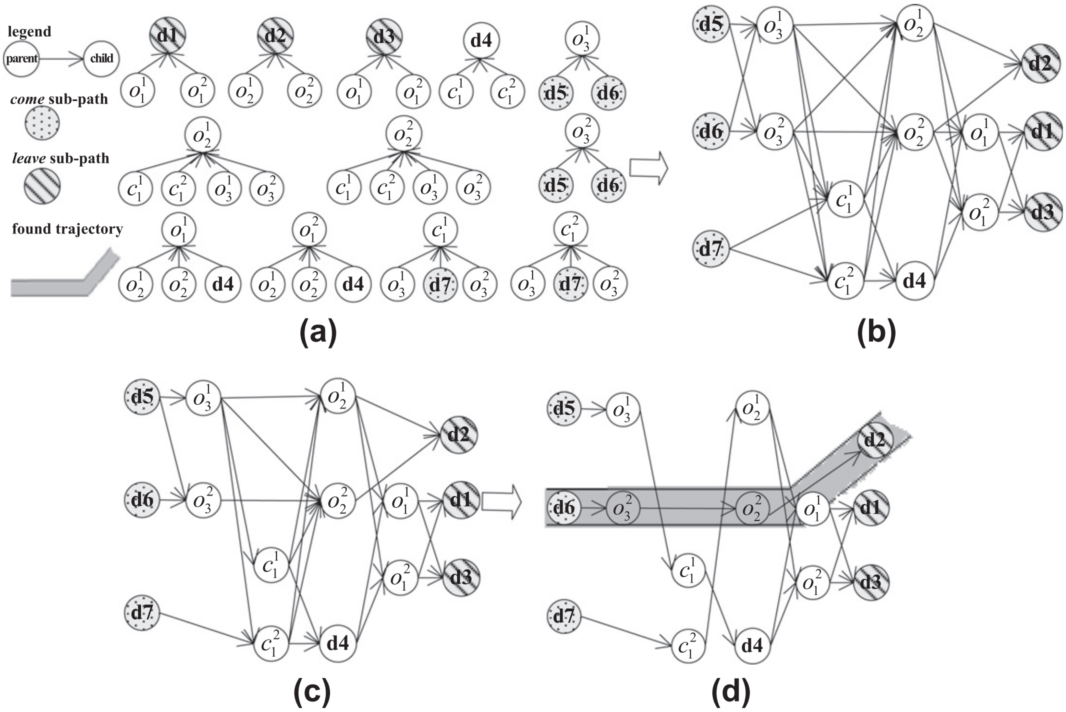
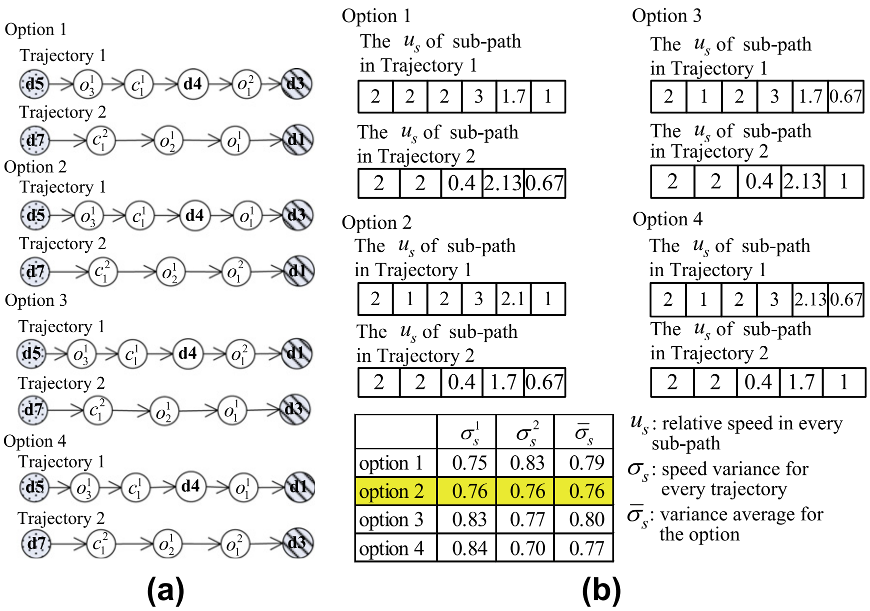


Fig. 7. Trajectory searching diagrammatic is presented to visualize the searching procedure. (a) Find parent–child relationships for every sub-path based on *physical reachability* of Ω among all of sub-path set (subprocedure **1** of Algorithm 4). (b) Build connecting diagram with parent–child relationships (subprocedure **2** of Algorithm 4). (c) Delete the relationship which is against the constraint (2), (subprocedure **3** of Algorithm 4). (d) Delete the relationship which is against the constraint (3), (subprocedure **4** and **5** of Algorithm 4).


Option 1
Trajectory 1
Trajectory 2
Option 2
Trajectory 1
Trajectory 2
Option 3
Trajectory 1
Trajectory 2
Option 4
Trajectory 1
Trajectory 2
(a)
Option 1
The u_s of sub-path in Trajectory 1
2 2 2 3 1.7 1
The u_s of sub-path in Trajectory 2
2 2 0.4 2.13 0.67
Option 2
The u_s of sub-path in Trajectory 1
2 1 2 3 2.1 1
The u_s of sub-path in Trajectory 2
2 2 0.4 1.7 0.67
Option 3
The u_s of sub-path in Trajectory 1
2 1 2 3 1.7 0.67
The u_s of sub-path in Trajectory 2
2 2 0.4 2.13 1
Option 4
The u_s of sub-path in Trajectory 1
2 1 2 3 2.13 0.67
The u_s of sub-path in Trajectory 2
2 2 0.4 1.7 1
u_s: relative speed in every sub-path
σ_s: speed variance for every trajectory
$\bar{\sigma}_s$: variance average for the option
(b)

	σ_s^1	σ_s^2	$\bar{\sigma}_s$
option 1	0.75	0.83	0.79
option 2	0.76	0.76	0.76
option 3	0.83	0.77	0.80
option 4	0.84	0.70	0.77

Fig. 8. Example to illustrate MPBT Algorithm. (a) Four possible trajectory options for above example, there are two trajectories in every option (sub-procedure 1 of Algorithm 5). (b) Relative *speed* calculation for the four trajectory options, and choose the option with the minimum average variance as the best trajectory combination (the shadowed row is the best one in this example).

5. Performance evaluation and experiments

A network of 20 TelosW [22] static wireless sensor nodes (Fig. 1b) are deployed throughout the 30 × 30 m floor (shown in Fig. 1a) workplace environment (in Georgia State University department of Computer Science). As earlier shown in the physical layout in Fig. 1a, the sensor nodes are deployed mainly in the hallways, key positions (e.g., entry points: nodes 103, 104; exit points: nodes 97, 104; positions with high motion activities in workday: nodes 118, 117, 102, 114, 100, 101), some rooms (e.g., printer room: node 92, kitchen: node 99, busy lab: node 93). These nodes are fixed on the ceiling and each of them is equipped with Panasonic AMN-31111 passive infrared motion sensor.

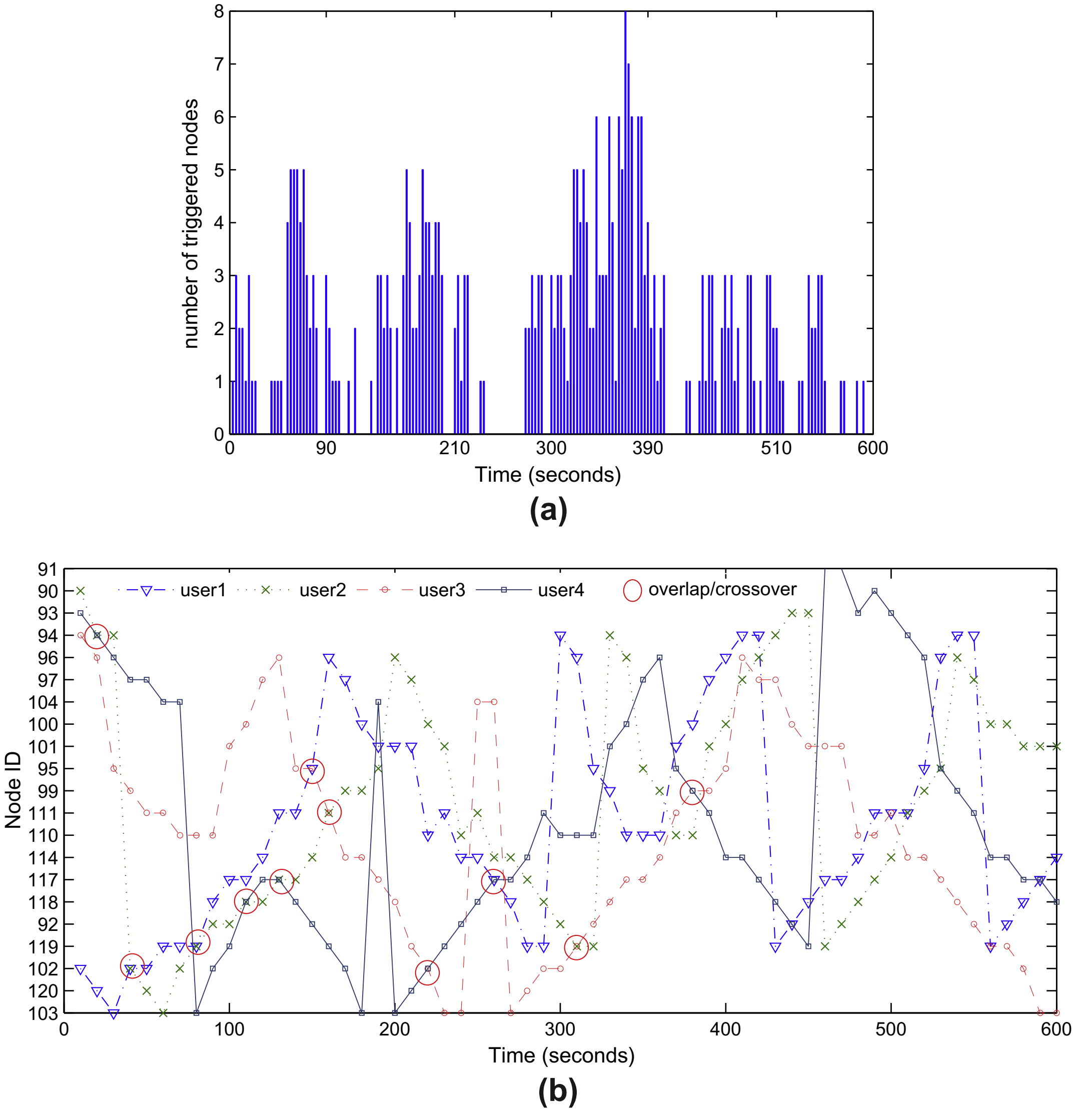


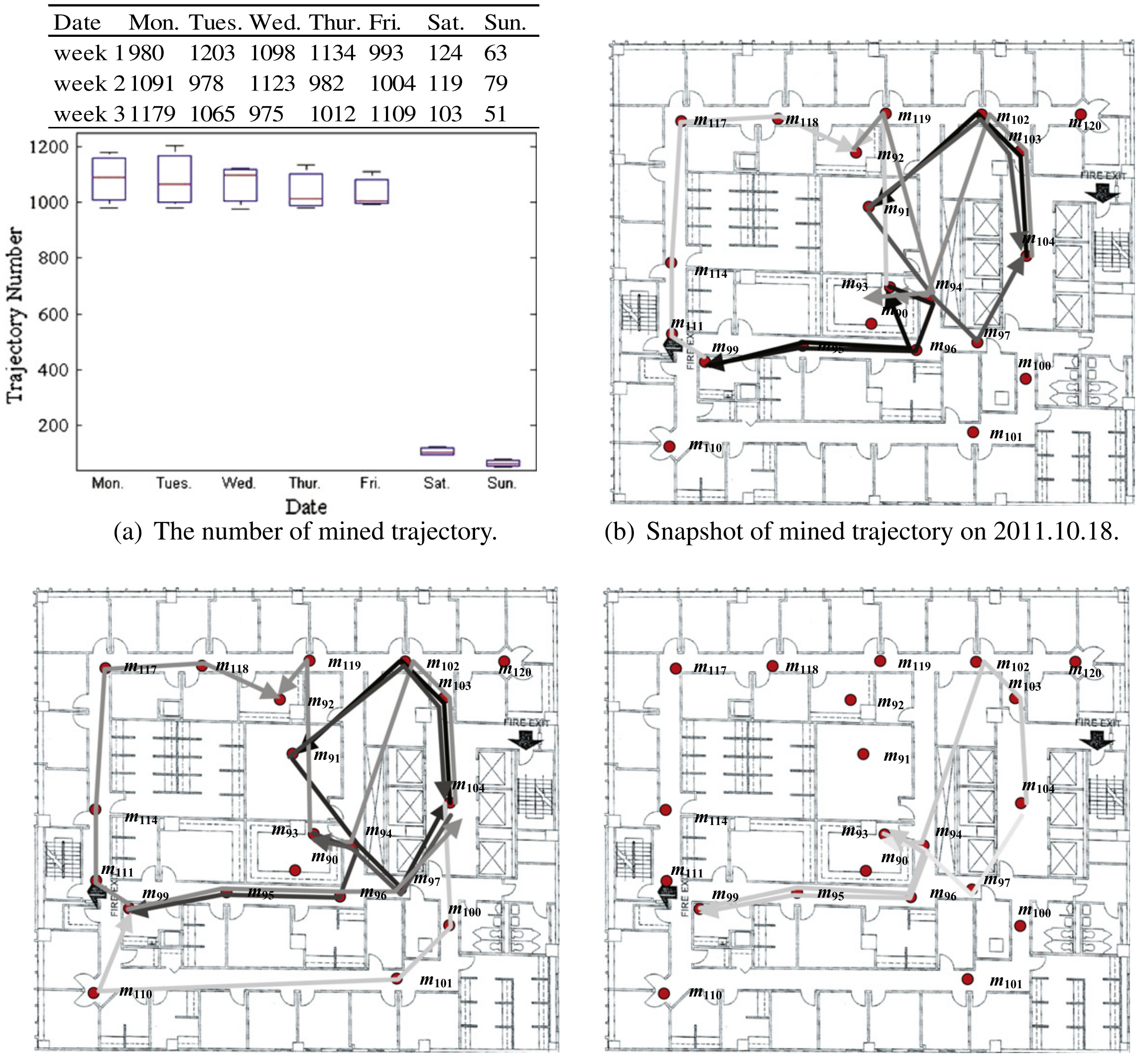
Fig. 9. One performance evaluation example has four users walking in scenario (1), and average path length of them is about 40. (a) Number of motion triggered nodes. (b) Ground truth motion trajectories of the four users, with the path overlap/crossover shown.

Table 1
Tracking accuracy results and comparison with *FindingHuMo* in [23].

Experiment no.	Scenarios	Users number	Average path length	*MiningTraMo*	*FindingHuMo*
#1	(1)	4	20	0.950	0.900
#2	(1)	4	65	0.954	0.861
#3	(1)	8	20	0.900	0.857
#4	(1)	8	65	0.919	0.734
#5	(2)	4	24	0.938	0.724
#6	(2)	4	63	0.944	0.692
#7	(2)	8	24	0.896	0.779
#8	(2)	8	63	0.897	0.561
#9	(3)	4	21	0.929	0.716
#10	(3)	4	67	0.933	0.645
#11	(3)	8	21	0.845	0.612
#12	(3)	8	67	0.847	0.538

5.1. Performance evaluation with ground truth data

Experiment setup: Firstly, we need design some examples that can generate ground truth and reflect the typical situation when user walks in a workplace. There are some volunteers (users) involving in and every user generates one trajectory. These users finish three scenarios: (1) users walk separately in random path, but overlap/crossover allowed; (2) users, two as a pair move together and one about 3 – 4 feet behind another, move in random paths with overlap/crossover allowed; (3) users walk in random paths with overlap/crossover allowed, but sometimes they can merge and walk together for a while, then separate. We design



Date	Mon.	Tues.	Wed.	Thur.	Fri.	Sat.	Sun.
week 1	980	1203	1098	1134	993	124	63
week 2	1091	978	1123	982	1004	119	79
week 3	1179	1065	975	1012	1109	103	51

(a) The number of mined trajectory.

(b) Snapshot of mined trajectory on 2011.10.18.

(c) Snapshot of mined trajectory on 2011.10.19.

(d) Snapshot of mined trajectory on 2011.10.23.

Fig. 10. Trajectory mining result of the experiment with real data.

12 examples to reflect the typical situation when user walks in a workplace, and one of these examples is shown in Fig. 9. In all of the three scenarios, every user carries a wireless device, which can send signal to database after being pressed, so as to label the time and device's ID, when he walks under any sensor, and there are four assistants will help to record the sensor's serial number at the same time. Then the ground truth of user position was recorded, to compare later with computed/mined user trajectories. *MiningTraMo* is compared here with *FindingHuMo* in [23].

Performance evaluation: The trajectory tracking accuracy results of *MiningTraMo* are list in Table 1, in which the tracking accuracy results of *FindingHuMo* in [23] are also listed. Because there is no training data for the calculation of HMM model λ that is required for *FindingHuMo*, we set λ based on our priori knowledge. For every one of the three scenarios, there are four and eight users respectively, as to present how the count of users, who walk in the Smart Environment simultaneously, influences tracking accuracy. In these examples, users' trajectories are different from each other and they are required to walk a short trajectory and a long one respectively. So we can evaluate how the path length of trajectory impact tracking accuracy. There are some key observations made from the comparative performance analysis. (i) From #1 vs #2, #3 vs #4, #5 vs #6, #7 vs #8, #9 vs #10 and #11 vs #12, it is clearly that path length almost have no influence on tracking performance of *MiningTraMo*. (ii) #1 vs #3, #2 vs #4, #5 vs #7, #6 vs #8, #9 vs #11 and #10 vs #12 tell us that more users leads to less accuracy. (iii) Scenario (2) and (3) indicate users have higher probability to overlap/crossover, that explain why the tracking accuracy of those experiments of scenario (1) is higher than the others of (2) and (3). (iv) *MiningTraMo* has higher tracking accuracy than *FindingHuMo* without training data for HMM.

5.2. Mining experiment with history data

The second kind experiment, which focus on the knowledge extracted from long history sensor data based on our framework, uses 3 weeks data from 2011.10.10.06:00 to 2011.10.31:06:00, and we combine motion data in a window of 1000 ms, and set up path-length-parameter $\xi = 2$. The result of mining work for this experiment is shown in Fig. 10a–d. The number of valid trajectory of the seven days is presented in Fig. 10a. It is noticed that there are much more valid trajectories during the week days than in the weekends. Fig. 10b and c respectively are the snapshots of these valid trajectories of 2011.10.18 and 2011.10.19, when both are working time for this Smart Workplace. Fig. 10d is the snapshot of 2011.10.23, the Sunday of that week. The different colors of these polylines in Fig. 10b–d denote these trajectories. These are used to indicate the different count of the mined trajectories with same classification of trajectories whose u, t_i and t_j may be different, but share the same triggered sensor set. Darker polyline means larger count and brighter one means smaller count. In order to present a clear snapshot, we ignore those trajectories whose count is less than 10. Because the Smart Environment in this experiment is an office place for research, the working days are busy for most of professors and students, however, there are still some hard-working people coming to their offices during the weekend. This is clearly proven by Fig. 10a. From Fig. 10b–d, it is impressive to know that there is only a minor difference between the trajectories of 2011.10.18 and these of 2011.10.19. However, the trajectories of 2011.10.23 are quite different from those of the working day, no matter what, considering the number or classification.

6. Discussion and conclusion

This paper discusses the issue of mining meaningful and useful trajectories from the data set collected from anonymous binary motion sensors, which are deployed in a Smart Environment. Based on the designed *MiningTraMo*, valuable and trust-worthy motion trajectories can be extracted from the data collected from simple non-invasive sensors. The performance evaluation is demonstrated with results from designed experiments in a smart workplace. In the future, our work will focus on: (i) designing some experiments, which have more people involving in and require longer walking time, to collect much more ground truth data, (ii) optimizing the trajectory mining framework for lower complexity using hash-based technique same as in [5], (iii) introducing multiple objective optimization algorithm to improve the trajectory searching strategy, and (iv) analyzing the behavior pattern of users in the Smart Environment and the functional relationship among the different rooms in Smart Environment.

Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grant No. 61004112.

References

[1] F. Giannotti, M. Nanni, D. Pedreschi, F. Pinelli, Trajectory pattern mining, in: Proceedings of the13th ACM Sigkdd International Conference on Knowledge Discovery and Data Mining (SIGKDD'07), ACM, San Jose, CA, 2007, pp. 330–339.

[2] P. Rashidi, D.J. Cook, L.B. Holder, M. Schmitter-Edgecombe, Discovering activities to recognize and track in a smart environment, IEEE Transactions on Knowledge and Data Engineering 23 (4) (2011) 527–539.

[3] Y. Busnel, L. Querzoni, R. Baldoni, M. Bertier, A.M. Kermarrec, Analysis of deterministic tracking of multiple objects using a binary sensor network, ACM Transactions on Sensor Networks 8 (1) (2011).

[4] D. Jung, T. Teixeira, A. Savvides, Towards cooperative localization of wearable sensors using accelerometers and cameras, in: INFOCOM 2010, 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15–19 March 2010, San Diego, CA, USA, 2010, pp. 2330–2338.

[5] E.T. Wang, A.L.P. Chen, A novel hash-based approach for mining frequent itemsets over data streams requiring less memory space, Data Mining and Knowledge Discovery 19 (1) (2009) 132–172.

[6] I. Wang, E. Johnson, J. Sokol, A multiple pairs shortest path algorithm, Transportation Science 39 (4) (2005) 465–476.

[7] D. De, S. Tang, W.Z. Song, D. Cook, S.K. Das, Actisen: activity-aware sensor network in smart environments, Journal of Pervasive and Mobile Computing (PMC) (2012).

[8] D. Cook, S. Das, Environments: Technology, Protocols and Applications, Wiley Series on Parallel and Distributed Computing, Wiley-Interscience, 2004.

[9] M. Youngblood, D. Cook, L. Holder, Seamlessly engineering a smart environment, in: International Conference on Systems, Man and Cybernetics, IEEE, Berlin, 2005, pp. 548–553.

[10] D.O. Olguin, B.N. Waber, T. Kim, A. Mohan, K. Ara, A. Pentland, Sensible organizations: technology and methodology for automatically measuring organizational behavior, IEEE Transactions on Systems Man and Cybernetics Part B – Cybernetics 39 (1) (2009) 43–55.

[11] J. Singh, U. Madhow, R. Kumar, S. Suri, R. Cagley, Tracking multiple targets-using binary-proximity sensors, in: Proceedings of the Sixth International Symposium on Information Processing in Sensor Networks, ACM, 2007, pp. 529–538.

[12] A. Razavian, Numerical object rings path planning algorithm, in: Proceedings of the 35th IEEE Conference on Decision and Control, IEEE, 1996, pp. 4406–4411.

[13] D. Zhang, Y. Liu, L. Ni, Rass: a real-time, accurate and scalable system for tracking transceiver-free objects, in: Pervasive Computing and Communications (PerCom'11), IEEE, 2011, pp. 197–204.

[14] Z. Zhong, T. Zhu, D. Wang, T. He, Tracking with unreliable node sequences, in: IEEE INFOCOM Conference (INFOCOM'09), IEEE, 2009, pp. 197–204.

[15] Y. Gao, C. Li, G. Chen, L. Chen, X. Jiang, C. Chen, Bfpknn: an efficient *k*-nearest-neighbor search algorithm for historical moving object trajectories, in: 4th International Conference on Advances in Information Systems, ACM, 2006, pp. 70–79.

[16] R.H. Gueting, T. Behr, J. Xu, Efficient *k*-nearest neighbor search on moving object trajectories, VLDB Journal 19 (5) (2010) 687–714.

[17] J.G. Lee, J. Han, X. Li, H. Cheng, Mining discriminative patterns for classifying trajectories on road networks, IEEE Transactions on Knowledge and Data Engineering 23 (5) (2011) 713–726.

[18] Y. Huang, L. Zhang, P. Zhang, A framework for mining sequential patterns from spatio-temporal event data sets, IEEE Transactions on Knowledge and Data Engineering 20 (4) (2008) 433–448.

[19] G. Yavas, D. Katsaros, O. Ulusoy, Y. Manolopoulos, A data mining approach for location prediction in mobile environments, Data and Knowledge Engineering 54 (2) (2005) 121–146.

[20] P. Pathirana, A. Savkin, S. Jha, Location estimation and trajectory prediction for cellular networks with mobile base stations, IEEE Transactions on Vehicular Technology 53 (6) (2004) 1903–1913.

[21] S. Schroedl, K. Wagstaff, S. Rogers, P. Langley, C. Wilson, Mining GPS traces for map refinement, Data Mining and Knowledge Discovery 9 (1) (2004) 59–87.

[22] G. Lu, D. De, M. Xu, W.Z. Song, B. Shirazi, Telosw: enabling ultra-low power wake-on sensor network, in: Seventh International Conference on Networked Sensing Systems (INSS'10), IEEE, 2010.

[23] D. De, W.Z. Song, M. Xu, C. Wang, D. Cook, X. Huo, Findinghumo: real-time tracking of motion trajectories from anonymous binary sensing in smart environments, in: The 32nd International Conference on Distributed Computing Systems (ICDCS'12), 2012b.