Towards Translating Raw Indoor Positioning Data into Mobility Semantics

Li, Huan; Lu, Hua; Chen, Gang; Chen, Ke; Chen, Qinkuang; Shou, Lidan

Published in:
ACM Transactions on Data Science

Creative Commons License
CC BY 4.0

Publication date:
2020

Document Version
Accepted author manuscript, peer reviewed version

Link to publication from Aalborg University

Citation for published version (APA):
Li, H., Lu, H., Chen, G., Chen, K., Chen, Q., & Shou, L. (Accepted/In press). Towards Translating Raw Indoor Positioning Data into Mobility Semantics. *ACM Transactions on Data Science*.

Towards Translating Raw Indoor Positioning Data into Mobility Semantics

HUAN LI and HUA LU[*], Department of Computer Science, Aalborg University, Denmark
GANG CHEN, KE CHEN, QINKUANG CHEN, and LIDAN SHOU[†], Department of Computer Science, Zhejiang University, China

Indoor mobility analyses are increasingly interesting due to the rapid growth of raw indoor positioning data obtained from IoT infrastructure. However, high-level analyses are still in urgent need of a concise but semantics-oriented representation of the mobility implied by the raw data. This work studies the problem of translating raw indoor positioning data into *mobility semantics* that describe a moving object's mobility event (What) someplace (Where) at some time (When). The problem is non-trivial mainly because of the inherent errors in the uncertain, discrete raw data. We propose a three-layer framework to tackle the problem. In the *cleaning* layer, we design a cleaning method that eliminates positioning data errors by considering indoor mobility constraints. In the *annotation* layer, we propose a split-and-match approach to annotate mobility semantics on the cleaned data. The approach first employs a density based splitting method to divide positioning sequences into split snippets according to underlying mobility events, followed by a semantic matching method that makes proper annotations for split snippets. In the *complementing* layer, we devise an inference method that makes use of the indoor topology and the mobility semantics already obtained to recover the missing mobility semantics. The extensive experiments demonstrate that our solution is efficient and effective on both real and synthetic data. For typical queries, our solution's resultant mobility semantics lead to more precise answers but incur less execution time than alternatives.

CCS Concepts: • **Information systems → Mobile information processing systems**; *Spatial-temporal systems*; *Data mining*.

Additional Key Words and Phrases: Indoor positioning data, semantic trajectory, mobility data mining

ACM Reference Format:
Huan Li, Hua Lu, Gang Chen, Ke Chen, Qinkuang Chen, and Lidan Shou. 2019. Towards Translating Raw Indoor Positioning Data into Mobility Semantics. *ACM Trans. Data Sci.* 1, 1, Article 1 (January 2019), 35 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

According to multiple studies [19, 21], people spend about 90% of their daily lives indoors. Human movements indoors are increasingly captured in indoor positioning data due to the recent advance in Internet-of-Things (IoT) wireless technologies [1] and high penetration of smartphones [2]. Analyzing indoor positioning data can reveal interesting findings otherwise hard to obtain, e.g.,

[*]H. Lu is the corresponding author.
[†]L. Shou is also with the State Key Laboratory of CAD&CG, Zhejiang University, China.

Authors' addresses: Huan Li, Hua Lu, Department of Computer Science, Aalborg University, Denmark, lihuan@cs.aau.dk, luhua@cs.aau.dk; Gang Chen, Ke Chen, Qinkuang Chen, Lidan Shou, Department of Computer Science, Zhejiang University, China, cg@zju.edu.cn, chenk@zju.edu.cn, chenqk@zju.edu.cn, should@zju.edu.cn.

2577-3224/2019/1-ART1 $15.00
https://doi.org/0000001.0000001

popular indoor locations [25, 26, 29], hot indoor routes [25, 34], useful indoor patterns [22], and insights for in-store marketing [13].

One class of indoor mobility analysis concerns semantics and asks questions like "Which combination of shops is most frequently visited by the shoppers in a mall?" or "List out the staff that meets at the security-sensitive regions after work hours and stays there for more than an hour." In order to answer such questions, we need to extract the mobility related semantics from the underlying indoor positioning data.

In this study, we use a type of data generated by indoor positioning systems based on IoT wireless technology (e.g., Wi-Fi or Beacon) [17, 18]. In particular, the data for one person (or moving object) contains a set of raw positioning records that are uncertain and discrete in nature. Each raw record $((x, y, f), t)$ means an object is observed at coordinate point (x, y) on floor f at time t. As the raw data does not give the needed semantics explicitly, we need to translate the raw data into an explicit, informative data representation. Inspired by the semantic trajectories [16, 33, 44], we use indoor mobility semantics exemplified as follows:

$$o_1 : (\textit{stay, Nike, 1:02pm-1:18pm}) \longrightarrow (\textit{pass, Adidas, 1:19pm-1:20pm})$$
$$\longrightarrow (\textit{stay, Cashier, 1:21pm-1:24pm})$$

Specifically, object o_1's movements are represented by a line of structured *mobility semantics*, which includes a mobility event annotation (a *stay* or *pass* event), a spatial annotation (a semantic region like *Nike* store), and a temporal annotation (a time period). We use *m-semantics*[1] to refer to such mobility semantics. As the annotations in m-semantics are related to indoor semantic regions and mobility events, m-semantics are considerably more comprehensible and useful for relevant application needs than the raw data [24].

However, translating raw indoor positioning data into m-semantics is still a challenging task mainly due to three reasons. First, the raw data obtained by wireless technology is extremely dirty because of unpredictable interferences of wireless signals [17, 18], especially when the infrastructure uses ordinary sensors, e.g., Wi-Fi access points, for network access [13]. Figure 1 depicts some Wi-Fi

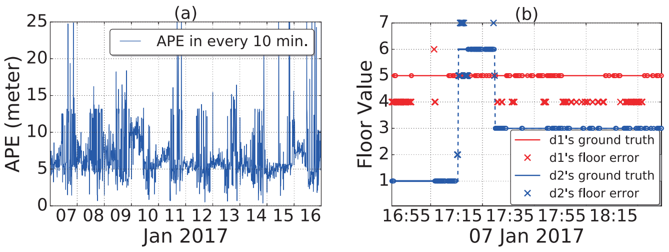


Fig. 1. Real-world Indoor Positioning Data Errors

based positioning data obtained in a real-world shopping mall in China. Referring to Figure 1(a), for a device sampled every 10 seconds, the average positioning error (APE) calculated every 10 minutes fluctuates significantly. Also, there are many false floor values reported for two example devices d1 and d2 as shown in Figure 1(b). Such data errors must be handled carefully before m-semantics can be extracted. Second, an indoor space usually has a relatively small extent but complex topology, leading to a compact distribution of semantic regions and complex movements under indoor topology. This makes the annotation more difficult. Existing outdoor annotation techniques make use of POI category [44] or assumptions such as human activity regularity [42]. However, in indoor spaces POIs of the same type often cluster together but object movements are

[1]Here, 'm' stands for 'mobility'.

quite random. Third, indoor positioning data is always discrete as mobile devices tend to turn off wireless signals for energy-saving needs [18]. It is non-trivial to obtain a complete sequence of m-semantics from discrete positioning data that at first glance suggests many possibilities.

Example 1. *Referring to the floorplan in Figure 2, some semantic regions are pre-defined such as S1 and hw-f. We illustrate an object's raw positioning data on the floorplan. Note that the location estimates represented as geometric points* (x, y) *are barely informative as no semantics can be recognized without the underlying floorplan provided. When interpreting the positioning data, due to the inherent errors and the layout of indoor regions, it is hard to make the spatial annotation (i.e., semantic region) for the object during the time period 9:05am-9:15am. Also, it is hard to construct m-semantics between the two presences in hallways hw-b at 9:16am and hw-d at 9:19am as no straightforward information is available by the discrete indoor positioning records.*

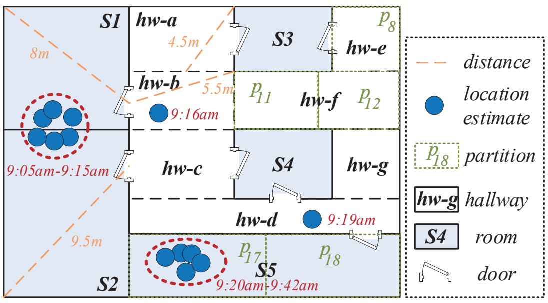


Fig. 2. Example of Indoor Floorplan

To address the aforementioned challenges, we propose a three-layer framework for constructing m-semantics. In the framework, the ultimate task is decomposed and accomplished by three functional layers in a chained manner. Each layer is equipped with corresponding applicable techniques to facilitate data processing. The *cleaning* layer considers the characteristics of indoor mobility constraints and cleans the raw positioning sequences. Subsequently, in the *annotation* layer, each positioning sequence is split into a number of snippets according to the underlying mobility event. The snippets are then translated into m-semantics by semantic matching. Last, in the *complementing* layer, to recover the missing m-semantics in an m-semantics sequence, knowledge about indoor mobility is constructed from the m-semantics already obtained, and an inference based method is used to infer the missing m-semantics on top of the constructed knowledge. Consequently, each object's m-semantics sequence is complemented. A prototype of the framework named TRIPS [24] has been built and released online.

To sum up, we make the following contributions in this paper.

- We formulate the problem of translating raw indoor positioning data into m-semantics and propose a three-layer framework to solve the problem (Section 2).
- We design a cleaning method that eliminates indoor positioning data errors based on indoor mobility constraints (Section 3).
- We devise a split-and-match approach to annotate m-semantics. It includes a density based method that splits the positioning sequence according to the underlying mobility event and a semantic matching method that makes annotations for the split snippets (Section 4).
- We propose an inference method to recover the missing m-semantics with the help of knowledge about indoor mobility, indoor topology, and m-semantics already obtained (Section 5).
- We conduct extensive experiments on both real and synthetic data to evaluate the efficiency and effectiveness of our proposals. (Section 6).

In addition, Section 7 reviews the related work and Section 8 concludes the paper.

2 PRELIMINARIES

Table 1 lists the notations used throughout this paper.

Table 1. Notations

Symbol	Meaning
$\theta = (l, t)$	a positioning record
$\Theta_o = \{(l_1, t_1), \ldots, (l_n, t_n)\}$	object o's positioning sequence
$r \in R$	an indoor (semantic) region, a spatial annotation
$\tau = [t_s, t_e]$	a temporal annotation
$\delta \in \{stay, pass\}$	an event annotation
$\lambda = (r, \tau, \delta)$	an indoor m-semantics
$\Lambda_o = \langle \lambda_1, \ldots, \lambda_m \rangle$	object o's ms-sequence
$PT = \langle r_1, \ldots, r_m \rangle$	region pattern of an ms-sequence
$\phi = r_s \rightarrow \ldots \rightarrow r_e$	an indoor candidate path
$dist_I(l_s, l_e)$	minimum indoor walking distance from location l_s to location l_e
$dist_{gr}(r_s, r_e)$	guaranteed reaching distance from region r_s to region r_e

2.1 Problem Formulation

In our setting, an indoor positioning system aperiodically reports a record $\theta(l, t)$ for an object o, where l is a location and t is a timestamp, meaning that object o's location is estimated to be l at time t. In most indoor positioning systems [17, 18], $\theta.l$ is represented as a triplet (x, y, f), i.e., a 2D point $(x, y) \in \mathbb{R}^2$ on a floor $f \in \mathbb{N}$. The positioning records are stored in an *indoor positioning table* (IPT), as exemplified in Table 2. Given an IPT and a particular time interval TI, we define an object's *indoor positioning sequence* (p-sequence) as follows.

Definition 1 (Indoor Positioning Sequence). *An object o's indoor positioning sequence over time interval* TI *is a time-ordered sequence* $\Theta_{o,\mathsf{TI}}$ *of positioning records of o, denoted as* $\Theta_{o,\mathsf{TI}} = \langle (l_1, t_1), \ldots, (l_n, t_n) \rangle$ *such that* $[t_1, t_n] \subseteq \mathsf{TI}$.

Table 2. Example of IPT

object	positioning record
o_1	$((2.5, 10.7, 1), t_1)$
o_2	$((5.1, 38.5, 4), t_1)$
o_1	$((2.3, 11.2, 2), t_4)$

Referring to Table 2, the object o_1's p-sequence over $\mathsf{TI} = [t_1, t_4]$ is $\langle((2.5, 10.7, 1), t_1), ((2.3, 11.2, 2), t_4)\rangle$. We formally give the definition of *mobility semantics* (m-semantics).

Definition 2 (Mobility Semantics). *An object o's mobility semantics is a triplet* $\lambda(r, \tau, \delta)$*, where the spatial annotation r is an indoor region, the temporal annotation* τ *is a time period, and the event annotation* δ *is a mobility event.*

The *indoor regions* serving as the spatial annotations are usually pre-defined with particular semantics by data analysts. For example, an indoor region can be a cashier or a shop in a mall. Essentially, an indoor space can be divided into a set of *indoor partitions* like rooms and hallways by walls and doors. For simplicity, we assume that each indoor region is composed of one or more such indoor partitions[2]. Referring to Figure 2, a semantic region *hw-e* corresponds to a single partition p_8 while another semantic region *S5* contains two partitions, i.e., p_{17} and p_{18}.

A *mobility event* refers to some interesting movement pattern. We introduce two generic patterns in our study, namely *stay* and *pass*.[3] A stay indicates that an object has been *staying in* a semantic region for a sufficiently long time for a particular purpose that is fulfilled in that region. For example, a stay can be that a user spent half an hour in a shoe shop, selecting and buying a pair of new shoes. It is noteworthy that since an indoor region itself usually carries some rich semantics determined by the use or planning of the space, combining the stay event with the corresponding region is very useful to disclose rich information about user behavior. For example, a stay at a restaurant usually means a person was dining while a stay at an office room indicates that a person was at work. In contrast, a pass tells that an object has *passed through* an indoor region but there is no particular purpose associated with that pass. For example, a user may pass several other shops before she reaches the shop where she buys shoes. The distinction between stay and pass is useful in pertinent indoor scenarios. For example, security managers may only be interested in people staying in a certain region, while the mall managers may want to know both numbers of staying and passing customers when analyzing the customer conversion rate in a region.

As to be introduced in Section 4.2.1, we design an *event identification function* ($\mathcal{E}$-function for short) to differentiate a stay and a pass. The $\mathcal{E}$-function is a learning based model and semi-supervised with the spatiotemporal features extracted from the user-recognized mobility events. An m-semantics annotated with a stay (pass) event is called a stay (pass) m-semantics and denoted by $\lambda^{\shortparallel}$ ($\lambda^{\triangleright}$), and its associated region is a stay (pass) region denoted as $r^{\shortparallel}$ ($r^{\triangleright}$).

Next, we define m-semantics sequence (ms-sequence).

Definition 3 (M-semantics Sequence). *An indoor object o's* m-semantics sequence *over time interval* TI *is a time-ordered sequence $\Lambda_{o,\mathsf{TI}}$ of o's m-semantics:* $\forall \lambda_i, \lambda_j \in \Lambda_{o,\mathsf{TI}}$, $\lambda_i.\tau \subseteq \mathsf{TI}$, $\lambda_j.\tau \subseteq \mathsf{TI}$, $\lambda_i.\tau \cap \lambda_j.\tau = \varnothing$.

When time context is clear, we use Θ_o and Λ_o to denote object o's p-sequence and ms-sequence, respectively. We formulate our research problem as below.

Research Problem (Indoor M-semantics Construction). *Given an IPT and a full set of indoor semantic regions, the* indoor m-semantics construction *translates each object o's p-sequence $\Theta_o = \langle \theta_1, \ldots, \theta_n \rangle$ in IPT into the most-likely ms-sequence $\Lambda_o = \langle \lambda_1, \ldots, \lambda_m \rangle$.*

Constructing ms-sequences provides an intuitive, concise way to understand a moving object's general indoor behaviors. It serves as the foundation of multiple high-level mobility analyses. However, the major challenge to the construction is that the input data is of very low quality and only provides very limited information. To this end, we propose a three-layer framework that progressively improves the data quality and constructs the m-semantics.

2.2 Framework Overview

As illustrated in Figure 3, our framework takes each object's p-sequence from the IPT as input and exports the corresponding ms-sequence. The data is processed through three functional layers.
Cleaning Layer handles the data errors in the p-sequence of each object. It conducts the data cleaning by considering the indoor mobility constraints captured in a distance-aware model.
Annotation Layer first uses a density based method to split each p-sequence into a number of snippets, and then translates each snippet into a number of m-semantics by a semantic matching based on the $\mathcal{E}$-function and the *semantic region graph*.

[2]Assuming regions do not overlap, our techniques allow an indoor region to involve (parts of) an indoor partition.
[3]The outdoor patterns that carry similar meanings are known as *stop* and *move* in other literatures [4, 33, 44].

Complementing Layer recovers the missing m-semantics for each original ms-sequence from the annotation layer. By making use of all the m-semantics already annotated, the *mobility knowledge* is obtained by a knowledge construction. Subsequently, each ms-sequence is complemented with a number of missing m-semantics by using an m-semantics inference with mobility knowledge.

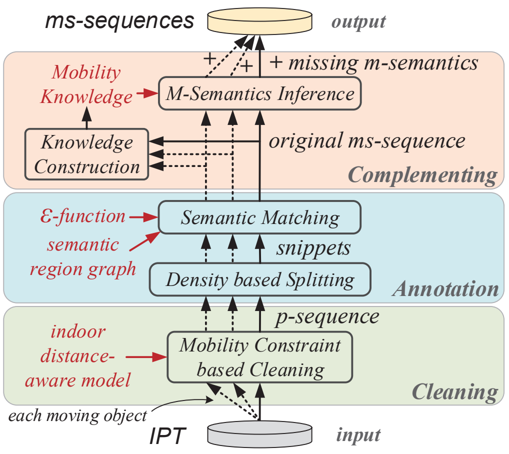


Fig. 3. The Construction Framework

The translation framework has been implemented in our prototype system TRIPS [24][4].

3 RAW POSITIONING DATA CLEANING

As shown in Figure 1, raw indoor positioning data contains inherent errors due to the limitations of the wireless based indoor positioning [18]. Typical errors are as follows.

1) *Random errors* are the small distortions from the true locations. They are caused by the imprecise measurement of wireless signals, which is easily influenced by factors such as temperature, humidity, and window opening or closing [23].
2) *Location outliers* are the significant deviations from the true locations. They occur when a mobile client suddenly fails to capture the signals from nearby transmitters. The location outliers discussed here are within the range of a floor.
3) *False floor values* are usually seen in multi-floor positioning systems. They occur in a case where a mobile client receives stronger signals from the transmitters on other floors.

These data errors impose serious problems on the subsequent processing of indoor m-semantics construction. It is necessary to identify and repair them to reduce their negative impacts.

Generally, indoor object movements should comply with underlying *mobility constraints*. For example, moving objects (usually people) cannot walk too fast indoors—a significant shift in the positioning data within a short time interval usually means a location outlier or a false floor value. Also, objects can move between indoor partitions only through doors or the like. Considering the moving speed between two positioning locations under indoor topology, we are able to identify a part of random errors that jump to other indoor partitions. We give an example in Figure 4. Suppose that an object o's p-sequence is $\langle(l_1, t_1), (l_2, t_2), (l_3, t_3)\rangle$, and l_1 at time t_1 is assumed to be valid. Given the maximum moving speed v_m and the specific indoor topology, o's position at time t_2 can only be inside the shaded part of the circle centered at l_1 with a radius $v_m \cdot (t_2 - t_1)$. As o's location report l_2 is outside the shaded part at time t_2, the reported location l_2 is an error.

[4]A demo video and relevant materials are available at https://longaspire.github.io/trips/.

When computing the object moving speed between two positioning locations, we use the *indoor distance-aware model* [28] to compute the *minimum indoor walking distance* (MIWD) between two indoor locations. In Figure 4, the MIWD from location l_1 to location l_3, denoted as $dist_I(l_1, l_3)$, is computed as the sum of two Euclidean distances $|l_1, d_1|$ and $|d_1, l_3|$, where d_1 is the door through which an object can reach l_3 from l_1.

Based on the MIWD that integrates both topological and geometrical mobility constraints, we identify the positioning data error by checking the indoor object speed. Formally, given a maximum speed v_m, for any two consecutive positioning records θ_i, θ_{i+1}, their in-between indoor speed $v = \frac{dist_I(\theta_i.l, \theta_{i+1}.l)}{\theta_{i+1}.t - \theta_i.t}$ should not exceed v_m. In other words, assuming we check a p-sequence in the forward direction, a record θ_i that violates the aforementioned speed constraint should be invalid if the precedent record θ_{i-1} has already been determined as valid.

When an invalid record θ_i is identified, the error may occur at its floor part (i.e., false floor value) and/or its 2D point part (i.e., location outlier or random error). We repair it in two steps. The first step repairs a potential mistake of floor value. If θ_i's floor value is different from the previous valid record θ_p's ($p \leq i-1$), we modify θ_i's to the same as θ_p's. Such modification comes into effect if the violation of speed constraint no longer occurs; otherwise, the potential error is still in the 2D point part. The second step repairs the wrong location estimate by interpolating a new one: For the current record θ_i, we find the previous and next valid positioning record as θ_p and θ_s ($s \geq i+1$), respectively. As no information is provided about the object movement between the two valid records, we assume that the object moved along the shortest indoor path between locations $\theta_p.l$ and $\theta_s.l$ at a constant speed. This assumption simplifies the interpolation but still complies with the mobility constraints. Consequently, given the timestamp $\theta_i.t$, the corresponding new location estimate is interpolated as a location l on the shortest indoor path from $\theta_p.l$ to $\theta_s.l$ such that $dist_I(\theta_p.l, l) = \frac{\theta_i.t - \theta_p.t}{\theta_s.t - \theta_p.t} \cdot dist_I(\theta_p.l, \theta_s.l)$. Also referring to Figure 4, location estimate l_2 (l_3) is determined as invalid (valid) based on the speed checking. Moreover, we should further repair l_2 by the second step as the error is in its 2D point part. Precisely, the possible position at time t_2 should be inside the blue shaded region provided that l_1 and l_3 have been determined as valid. For simplicity, we interpolate the new estimate l_2' for o at time t_2 on the shortest indoor path from l_1 and l_3 having $dist_I(l_1, l_2')/dist_I(l_2', l_3) = (t_2 - t_1)/(t_3 - t_2)$. It can be proved that l_2' must be inside the blue shaded region. We omit the details due to the page limit.

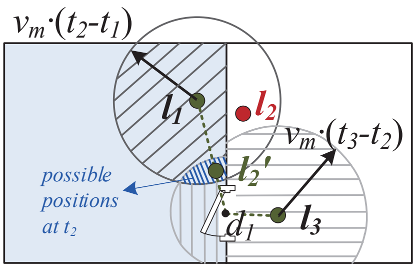


Fig. 4. Example of Cleaning Raw Indoor Positioning Data

The forward version of cleaning is formalized in Algorithm 1. For simplicity of presentation, this version assumes that the first record of the p-sequence is valid. Nevertheless, our cleaning can start with any valid record in the middle such that the remaining ones can be checked forward and backward from the start record. In particular, a record can be selected as the start record if it has both the speeds to its precedent and subsequent records not exceed the threshold v_m. Similar to the forward cleaning in Algorithm 1, a backward cleaning can be performed to handle the records before the start record selected.

Algorithm 1: Mobility Constraint based Cleaning in the Forward Direction

```
Input: p-sequence $\Theta_o$, maximum moving speed $v_m$.
Output: cleaned p-sequence $\Theta'_o$.
1  time-ordered sequence $\mathcal{A}_\theta \longleftarrow \langle\rangle$
2  current valid positioning record $\hat{\theta} \longleftarrow head(\Theta_o)$
3  for each positioning record $\theta \in \Theta_o \setminus head(\Theta_o)$ do
4      $valid \longleftarrow$ True
5      if $ValidSpeed(\hat{\theta}, \theta)$ is False then
6          $(x, y, f) \longleftarrow \theta.l$; $(\hat{x}, \hat{y}, \hat{f}) \longleftarrow \hat{\theta}.l$; $\theta.l \longleftarrow (x, y, \hat{f})$
7          if $ValidSpeed(\hat{\theta}, \theta)$ is False then
8              add $\theta$ to $\mathcal{A}_\theta$; $valid \longleftarrow$ False
9      if $valid$ then
10         $\mathcal{A}'_\theta \longleftarrow Interpolation(\hat{\theta}, \mathcal{A}_\theta, \theta)$
11         add all positioning records in $\mathcal{A}'_\theta$ to $\Theta'_o$
12         $\mathcal{A}_\theta \longleftarrow \langle\rangle$; $\hat{\theta} \longleftarrow \theta$
13 add $\hat{\theta}$ to $\Theta'_o$
14 return $\Theta'_o$
15 Function $ValidSpeed(\hat{\theta}, \theta)$
16     $v \longleftarrow dist_I(\hat{\theta}.l, \theta.l)$ / $(\theta.t$ - $\hat{\theta}.t)$ // compute the MIWD
17     if $v \leq v_m$ then return True else return False
18 Function $Interpolation(\theta_p, \mathcal{A}_\theta, \theta_s)$
19     $\mathcal{A}'_\theta \longleftarrow \langle\rangle$; add $\theta_p$ to $\mathcal{A}'_\theta$
20     compute a shortest indoor path $pa$ from $\theta_p.l$ to $\theta_s.l$
21     for each positioning record $\theta' \in \mathcal{A}_\theta$ do
22         find a location $l$ on $pa$ having $dist_I(\theta_p.l, l) = \frac{\theta'.t-\theta_p.t}{\theta_s.t-\theta_p.t} \cdot dist_I(\theta_p.l, \theta_s.l)$
23         $\theta'.l \longleftarrow l$; add $\theta'$ to $\mathcal{A}'_\theta$
24     return $\mathcal{A}'_\theta$
```

4 MOBILITY SEMANTICS ANNOTATION

In this section, we use a *split-and-match* approach to annotate the m-semantics on a cleaned p-sequence. First, we split the p-sequence into a number of *snippets*, each corresponding to an underlying mobility event (i.e., stay or pass introduced in Section 2.1). Next, for each split snippet, we match its m-semantics by making the three annotations (see Definition 2). The overall process is formalized in Algorithm 2. We give the density based splitting method (called in line 2) in Section 4.1, and elaborate on the semantic matching method (called in line 5) in Section 4.2.

4.1 Density Based Splitting

In our observation, the positioning records associated with a stay event are likely to have their location estimates and timestamps packed together. Inspired by this, we propose a density based clustering method to find a number of clusters and split the p-sequence based on these clusters. In particular, the positioning records contained in a cluster form a dense snippet, and those consecutive records between two clusters form a non-dense snippet. *ST-DBSCAN* [8] is a competent algorithm to cluster the sequential data instances according to the spatial and temporal attributes. It requires three parameters: 1) ϵ_s is a distance threshold for spatial attributes; 2) ϵ_t is a distance threshold

Algorithm 2: Split-and-Match Annotation

```
Input: p-sequence $\Theta_o$, event identification function $\mathcal{E}$, semantic region graph $G_R$.
Output: time-ordered sequence of m-semantics $\Lambda_o$.
1 time-ordered sequence $\Lambda_o \longleftarrow \langle\rangle$
2 $\mathcal{A}_{snpt} \longleftarrow DensityBasedSplitting(\Theta_o)$
3 for each snippet $\Theta_o^*$ in $\mathcal{A}_{snpt}$ do
4     $\Lambda_o^* \longleftarrow SemanticMatching(\Theta_o^*, \mathcal{E}, G_R)$
5     $\Lambda_o \longleftarrow \Lambda_o \cup \Lambda_o^*$
6 return $\Lambda_o$
```

for temporal attributes; 3) pt_m is a numerical threshold. A cluster is formed only if it contains at least pt_m data instances and any instance in it is within the spatial distance ϵ_s *and also* within the temporal distance ϵ_t to another instance in it. Note that the instances in a formed cluster are continuous on the time attribute. Compared to the traditional speed based methods like SMoT [4] that use a hard-coded speed threshold to distinguish stay and pass events, our density based method takes a more comprehensive consideration of the spatiotemporal data distribution of different mobility events. We experimentally compare different splitting methods in Section 6.1.1.

To enable the density based splitting on p-sequence, we extend ST-DBSCAN in three aspects. First, we introduce the MIWD as the distance metric for spatial attributes with respect to the indoor topology. Second, we build the parameter pt_m adaptively rather than using a constant value. In particular, pt_m at time t_i is associated with the *local* sampling rate within the time window $[t_i - \epsilon_t, t_i + \epsilon_t]$. If the local sampling rate is currently low, i.e., only a few positioning records were observed within the time window, we use a small pt_m. In contrast, a large pt_m is used when the local sampling rate is high. This way makes it flexible to form clusters in the context of dynamically changing sampling rates. Third, we introduce two parameters, namely *tolerate time span* Δ_t and *tolerate spatial distance* Δ_s, to avoid small, fragmentary dense snippets to be formed. Formally, any two dense snippets $\langle\theta_i, \ldots, \theta_j\rangle$ and $\langle\theta_k, \ldots, \theta_l\rangle$ are merged if 1) $\theta_k.t - \theta_j.t \leq \Delta_t$, and 2) $\exists s \in [i, j], \exists t \in [k, l], dist_I(\theta_s.l, \theta_t.l) \leq \Delta_s$. The pseudo-codes of the procedure are given in Algorithm 4 in Appendix A.1.

Example 2. *Referring to the splitting in Figure 5, the positioning records in a cluster formed within the time period 9:05am-9:15am are captured as a dense snippet D_1. Also, three captured dense snippets within 9:20am-9:42am are merged together according to the condition defined on Δ_s and Δ_t, resulting in another dense snippet D_3. Between D_1 and D_3, there are two non-clustered records within 9:16am-9:19am; they form a non-dense snippet D_2. As a result, the p-sequence is split into three parts.*

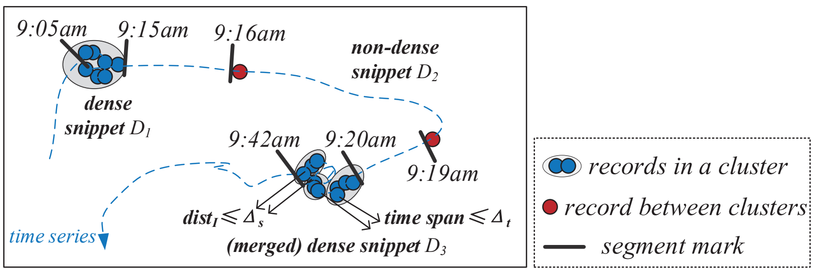


Fig. 5. Example of Density based Splitting on a P-sequence

The density information provides a good reference for splitting a p-sequence. However, it is not sufficient to directly regard a dense snippet as a stay and a non-dense one as a pass. Suppose that an object moves steadily and reports its location at a high frequency. Although the reported locations (and timestamps) can be close to each other and satisfy the clustering conditions, it would be wrong to consider the object is staying in a place. To verify if a snippet corresponds to a true stay event, more information, e.g., total travel distance and location estimate variance, needs to be extracted from its containing positioning records and be checked further. To this end, we introduce the event identification technique in Section 4.2.1.

4.2 Semantic Matching

4.2.1 Event Identification Function. To determine the underlying mobility event associated with a snippet, we design an event identification function ($\mathcal{E}$-function) based on a semi-supervised learning model. In particular, each snippet (i.e., a segment of raw records) is first represented as a mobility feature vector. Features are extracted from the following aspects.

- *Dense Level.* As the positioning records of stay events usually fall in the formed clusters, it is useful to indicate whether the snippet is dense or not. We represent the dense level of the snippet as a binary value.
- *Variance of Location Estimates.* As the variance of location estimates is usually small in stay events but large in pass events, we compute the location variance of the snippet as a feature.
- *Sampling Conditions.* Since the indoor positioning data is usually sparser when the object (wireless device) is moving, we compute the record number and the sampling rate in the snippet as two features.
- *Covering Range.* As a larger covering range tends to be a pass event, we compute the area and the longest distance from the centrum to a vertex of the geometric shape (i.e., convex hull or its simplified minimum boundary rectangle) that covers all location estimates of the snippet.
- *Overlapping Regions.* Considering the effect of different indoor places, we compute the dummy encoded IDs of the neighboring regions that contain or intersect with the covering range, together with each such region's relevant record number. Four nearest regions are considered and the empty dimensions (happened when there're no enough overlapping regions) are filled with zero.
- *Walking Distance.* As a snippet with a larger walking distance tends to be a pass event, we compute the sum and average of the MIWDs between every two consecutive location estimates as two features.
- *Walking Speed.* As a snippet with a larger walking speed tends to be a pass event, we compute the maximum, minimum and average of the instant speeds between every two consecutive records as three features.
- *Number of Turns.* Studies reveal that people only make a very small number of turns when they are walking indoors [34]. Therefore, we compute the ratio between the number of turns and the walking distance to help identify pass events. Particularly, for any location estimate $\theta_i.l$, a turn is identified if the angle between the line from $\theta_{i-1}.l$ to $\theta_i.l$ and the line from $\theta_i.l$ to $\theta_{i+1}.l$ exceeds 90 degrees.

The logistic regression model [3] is employed to classify stop and pass events. To train the model, a feature set is extracted from the snippets labeled with *stop* or *pass* and then normalized. Logistic regression is efficient to train and it produces a probability value of being one class on the sigmoid curve. Such an advantage enables us to use a co-training method [9] that constructs additional labeled data to overcome the lack of training data. Particularly, the predicted snippets with very high or very low probability values can be added as positive and negative samples, respectively.

Consequently, given a snippet Θ_o^*, its mobility event returned by $\mathcal{E}(\Theta_o^*)$ corresponds to either *stop* or *pass* predicted by the classifier. Once the mobility event is determined, in Section 4.2.2 we determine the annotations for stay and pass m-semantics, especially the spatial annotation.

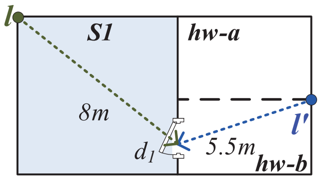


Fig. 6. GRD Example

4.2.2 Determining Annotations. For a snippet Θ_o^* that has $\mathcal{E}(\Theta_o^*) = stay$, a stay m-semantics is generated with the temporal annotation made as $\tau = [head(\Theta_o^*).t, tail(\Theta_o^*).t]$. In contrast, Θ_o^* should be matched to one or more pass m-semantics as the corresponding object may move through different regions. In such a case, each positioning record $\theta \in \Theta_o^*$ is mapped to a pass m-semantics with its temporal annotation made as $\tau = [\theta.t, \theta.t]$.

Next, we need to make the spatial annotations for the aforementioned stay or pass m-semantics. In the following, we introduce a *semantic region graph* G_R that facilitates accessing the indoor semantic regions. Specifically, G_R is a labeled, directed graph represented in a five-tuple (R, E, G_{dist}, $R2P$, $P2R$):

1) R is the vertex set and each vertex in it corresponds to an indoor region r.
2) E is the edge set $\{\langle r_i, r_j, \mathbb{R}\rangle \mid r_i, r_j \in R\}$ and each directed edge gives the *guaranteed reaching distance* (GRD) from an indoor region r_i to another *directly connected*[5] indoor region r_j.
3) G_{dist} is the associated distance-aware model [28] that involves with indoor entities like doors and partitions.
4) $R2P : R \longrightarrow 2^P$ maps an indoor region (vertex) to the set of indoor partitions in G_{dist} it covers.
5) $P2R : P \longrightarrow R$ maps an indoor partition in G_{dist} to the indoor region that covers it.

The mappings $R2P$ and $P2R$ are maintained by two hash tables. They can be automatically constructed based on the topological computations between regions and partitions.

The **guaranteed reaching distance** (GRD) from a region r_i to a region r_j is defined as

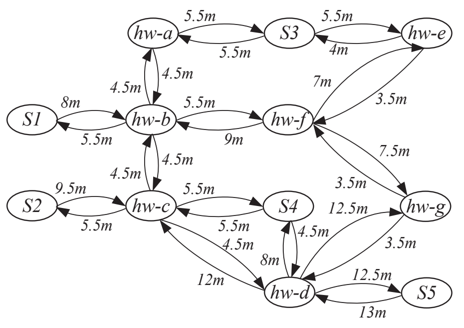
$$dist_{gr}(r_i, r_j) = \max_{l \in r_i, d \in P2D_{\sqsupset}(R2P(r_j))} dist_I(l, d) \tag{1}$$

where $R2P(r_j)$ gives the covering partitions of region r_j and the mapping $P2D_{\sqsupset}$ [28] gives the enterable doors for a given indoor partition. Generally, the GRD from r_i to r_j is the walking distance an indoor object needs to reach (an enterable door of) r_j from the farthest position in r_i. In other words, any object currently in r_i can reach r_j within the distance $dist_{gr}(r_i, r_j)$. Note that $dist_{gr}(r_i, r_j) \neq dist_{gr}(r_j, r_i)$. Referring to Figure 6, regions *S1* and *hw-b* are directly connected, while *S1* and *hw-a* are not as their in-between walking path must go through another region *hw-b*. Suppose that $l \in S1$ is the farthest position from the enterable door d_1 of *hw-b*. Thus, GRD from *S1* to *hw-b* equals to $dist_I(l, d_1) = 8m$. In contrast, GRD from *hw-b* to *S1* is $dist_I(l', d_1) = 5.5m$ where $l' \in hw\text{-}b$ is the farthest position from the enterable door d_1 of *S1*. The two GRDs indicate that it usually costs more time to walk out from a larger region like *S1* than from a smaller one like *hw-b*. This property of GRDs can be used to allocate the time periods for the regions that have been inferred in recovering the missing m-semantics. The details are to be given in Section 5.2.2.

Figure 7 gives the graph G_R corresponding to Figure 2. With the *R2P* and *P2R* mappings, region *hw-f* is mapped to two partitions that it contains (i.e., partitions p_{11} and p_{12} shown in Figure 2) and region *hw-e* is mapped to a single partition p_8. In contrast, partition p_{17} in Figure 2 is mapped to its covering region *S5*. Also, it can be seen that any object currently in *S2* can reach *hw-c* by a distance $dist_{gr}(S2, hw\text{-}c) = 9.5m$, consistent with the precise drawing in Figure 2.

To speed up the spatial searching that involves the geometric location estimates and indoor regions, we index the regions' associated partitions in G_{dist} by an R-tree. When a given location

[5]Two regions are directly connected if an object can move from one to the other without getting into a third region.



R2P mapping

region	partition
hw-e	p_8
hw-f	$\{p_{11}, p_{12}\}$
...	...

P2R mapping

partition	region
p_{11}	*hw-f*
p_{17}	*S5*
p_{18}	*S5*
...	...

Fig. 7. Example of Semantic Region Graph G_R

estimate's intersected partition is found via the R-tree, the relevant covering region can be obtained via the *P2R* mapping defined in G_R. Note that these mappings allow users to specify the indoor regions without getting to the underlying spatial searching conducted on the level of indoor partitions. Consequently, we can use G_R to find the best-matched spatial annotation for an m-semantics according to its corresponding positioning record(s).

We differentiate the spatial annotation matching for pass and stay m-semantics. For a pass semantics and its corresponding positioning record θ, we simply consider the indoor region that contains $\theta.l$ as the spatial annotation. The consecutive pass m-semantics that are matched with the same spatial annotation should be merged together such that their time periods are combined. This helps reduce possible redundancy in the semantics.

The matching for stay m-semantics is more complex as it involves multiple location estimates. We find that an object staying in a place usually incurs small displacements between its consecutive location estimates. Therefore, an estimate fairly far away from its neighboring estimates should be affected by positioning errors and is less reliable. In this light, different from the matching methods that use the centroid of location estimates or voting results, we assign each estimate different importance in determining the stay region. We define *location estimate confidence* as follows.

Definition 4 (Location Estimate Confidence). *Given a snippet $\langle \theta_s, \ldots, \theta_e \rangle$ associated with a stay event, the confidence of a location estimate $\theta_i.l$ ($s \le i \le e$) is defined as*

$$conf^{(i)} = \left(\frac{\sum_{\theta_j \in \mathcal{N}^{(i)}} dist_I(\theta_i.l, \theta_j.l)}{|\mathcal{N}^{(i)}|}\right)^{-1} / Z \qquad (2)$$

where $\mathcal{N}^{(i)}$ is the set of θ_i's k nearest neighbor location estimates from other positioning records in the snippet and Z is a normalization parameter making the maximum confidence be 1.

In the definition, we evaluate each estimate's confidence by making use of its average MIWDs to the neighboring estimates. Enabled by the evaluated confidence, we compute each estimate $\theta_i.l$'s importance as $\beta^{(i)} = \frac{conf^{(i)}}{\sum_{j=s}^{e} conf^{(j)}}$ and infer the underlying stay position as $\hat{l} = \sum_{i=s}^{e} \beta^{(i)} \cdot \theta_i.l$. As a result, the spatial annotation for a stay m-semantics is made as a region that contains $\hat{l}$.

The whole procedure of semantic matching is formalized in Algorithm 5 in Appendix A.2.

Example 3. *Referring to Figure 5, suppose that the split snippets D_1, D_2, and D_3 have been determined by the $\mathcal{E}$-function as stay, pass, and stay, respectively. Referring to the snippet D_1 within 9:05am-9:15am, its included location estimates are pointed out by a dashed circle in Figure 2. Based on the spatial annotation matching introduced above, D_1 is mapped to an m-semantics (S1, 9:05am-9:15am, stay). Similarly, D_3 within 9:20am-9:42am is translated into (S5, 9:20am-9:42am, stay) as can* *be referred in Figure 2. Besides, the two records in D_2 (within 9:16am-9:19am) is translated into two pass m-semantics, i.e., (hw-b, 9:16am-9:16am, pass) and (hw-d, 9:19am-9:19am, pass).*

5 COMPLEMENTING MS-SEQUENCES

An ms-sequence obtained from the annotation layer can still be incomplete in that some m-semantics are missing between the consecutive pass m-semantics. Next, we present an inference method to recover those missing m-semantics.

Multiple studies [22, 34] have discovered that people often make similar movements between two indoor destinations within a relatively small range, regardless of the walking purposes (e.g., to find something or just to look around somewhere). This inspires us to make use of such similar movements to infer the missing m-semantics between two relevant regions. Specifically, each stay region $r^{||}$ in an annotated m-semantics can be considered as an indoor destination, and those annotated pass m-semantics between two destinations can be grouped to capture the similar movements in-between. By further considering indoor mobility constraints, we are able to infer the unobserved movements given an ms-sequence we already annotated.

The complementing consists of two phases, as formalized in Algorithm 3. The first phase (line 2) constructs the mobility knowledge (i.e., similar movements) between each two stay regions (i.e., destinations). Using the constructed knowledge, the second phase (lines 3–5) infers the missing m-semantics for each ms-sequence. The two phases are detailed in Sections 5.1 and 5.2, respectively.

Algorithm 3: Inference based Complementing

```
  Input: set of ms-sequences $\mathcal{S}_\Lambda$, semantic region graph $G_R$.
  Output: set of complemented ms-sequences $\mathcal{S}'_\Lambda$.
1 set $\mathcal{S}'_\Lambda \longleftarrow \varnothing$
2 hash table $\mathcal{MK} \longleftarrow$ ConstructMobilityKnowledge($G_R$, $\mathcal{S}_\Lambda$)
3 for each original ms-sequence $\Lambda_o$ in $\mathcal{S}_\Lambda$ do
4     $\Lambda_o \longleftarrow$ MSemanticsInference($\Lambda_o$, $\mathcal{MK}$, $G_R$)
5     add $\Lambda_o$ to $\mathcal{S}'_\Lambda$
6 return $\mathcal{S}'_\Lambda$
```

5.1 Mobility Knowledge Construction

Given two stay regions $r^{||}_s, r^{||}_e$, the *mobility knowledge* about the similar movements from $r^{||}_s$ to $r^{||}_e$ consists of two parts. The first part is a set of candidate paths that accommodate those similar movements, and each path is represented as a sequence of directly connected regions in the semantic region graph G_R. The second part is the transition probabilities between the directly connected regions in the candidate paths. Next, we elaborate on constructing the candidate path set and the transition probabilities, respectively.

5.1.1 Candidate Path Set. We first define *indoor candidate path.*

Definition 5 (Indoor Candidate Path). *Given a start region $r^{||}_s$ and an end region $r^{||}_e$, a candidate path from $r^{||}_s$ to $r^{||}_e$ is a region sequence $\phi = r^{||}_s \rightarrow r^{\triangleright}_i \rightarrow \ldots \rightarrow r^{\triangleright}_j \rightarrow r^{||}_e$, each pass region $r^{\triangleright}_k$ $(i \le k \le j)$ contained in ϕ is* unique*, and each two consecutive regions in ϕ are* directly connected. *Path ϕ's* path length *$L(\phi)$ is given by*

$$dist_{gr}(r^{||}_s, r^{\triangleright}_i) + \sum_{k=i}^{j-1} dist_{gr}(r^{\triangleright}_k, r^{\triangleright}_{k+1}) + dist_{gr}(r^{\triangleright}_j, r^{||}_e) \quad (3)$$

where $dist_{gr}$ is the GRD kept in G_R.

The path length $L(\phi)$ is an upper bound of the minimum distance to ensure that any object can reach $r_e^{\shortparallel}$ from $r_s^{\shortparallel}$ along the path ϕ. The proof is given in Lemma B.1 in Appendix B.1.

To find a set Φ of candidate paths from $r_s^{\shortparallel}$ to $r_e^{\shortparallel}$, we perform an A*-Search [48] on G_R. As the number of such candidate paths can be very large, we use a path length threshold γ to filter out the paths whose length is extremely long as the similar movements are within a relatively small range [22]. The threshold γ should be determined according to the statistics on the path lengths between two stay regions. In our experiments, we set γ to the double of the length of the shortest candidate path from $r_s^{\shortparallel}$ to $r_e^{\shortparallel}$ as a path length in the experiments never exceeds this value.

Computing path lengths by using the sum of GRDs between directly connected regions can avoid more complex computations on the underlying distance-aware model G_{dist}. Moreover, it facilitates pruning the irrelevant regions (e.g., those whose sum of GRDs to $r_s^{\shortparallel}$ or $r_e^{\shortparallel}$ exceeds γ) when searching candidate paths on G_R.

Example 4. *Given the start region $r_s^{\shortparallel} = S1$ and end region $r_e^{\shortparallel} = S5$, the path length threshold γ is set to double of 29.5m, the shortest path length from S1 to S5. As a result, four indoor candidate paths can be searched from the G_R shown in Figure 7.*

$$S1 \overset{8m}{\rightarrow} \left\{ \begin{array}{l} 1.hw\text{-}b \overset{4.5m}{\rightarrow} hw\text{-}c \overset{4.5m}{\rightarrow} hw\text{-}d \\ 2.hw\text{-}b \overset{4.5m}{\rightarrow} hw\text{-}c \overset{5.5m}{\rightarrow} S4 \overset{4.5m}{\rightarrow} hw\text{-}d \\ 3.hw\text{-}b \overset{5.5m}{\rightarrow} hw\text{-}f \overset{7.5m}{\rightarrow} hw\text{-}g \overset{3.5m}{\rightarrow} hw\text{-}d \\ 4.hw\text{-}b \overset{4.5m}{\rightarrow} hw\text{-}a \overset{5.5m}{\rightarrow} S3 \overset{5.5m}{\rightarrow} hw\text{-}e \overset{3.5m}{\rightarrow} \\ \quad hw\text{-}f \overset{7.5m}{\rightarrow} hw\text{-}g \overset{3.5m}{\rightarrow} hw\text{-}d \end{array} \right\} \overset{12.5m}{\rightarrow} S5$$

5.1.2 Transition Probabilities. Next, we compute the transition probability between each two directly connected regions on a path from the candidate path set. Specifically, given a set of ms-sequences, a start region $r_s^{\shortparallel}$ and an end region $r_e^{\shortparallel}$, we first obtain their *region patterns* through the following steps.

1) For each ms-sequence, we find all its subsequence that starts with $\lambda_s^{\shortparallel}$ and ends with $\lambda_e^{\shortparallel}$, where the m-semantics $\lambda_s^{\shortparallel}$ ($\lambda_e^{\shortparallel}$) corresponds to the region $r_s^{\shortparallel}$ ($r_e^{\shortparallel}$).
2) For each such subsequence represented as $\langle \lambda_s^{\shortparallel}, \lambda_i^{\triangleright}, \ldots, \lambda_j^{\triangleright}, \lambda_e^{\shortparallel} \rangle$, we obtain its **region pattern** as $PT = \langle r_i^{\triangleright}, \ldots, r_j^{\triangleright} \rangle$[6].

We iterate through all ms-sequences and record each obtained region pattern and its count number in a hash table $\mathcal{H}_{PT}$. Subsequently, we compute the *transition probability* that an object leaves a region r_i for a directly connected region r_j as follows.

1) For each region pattern $PT = \langle r_i, \ldots, r_j \rangle$ in $\mathcal{H}_{PT}$, we find a subset Φ' of candidate path set Φ, in which all paths hold PT .
2) For each path $\phi \in \Phi'$, we compute its path length $L(\phi)$ according to Equation 3. As moving objects tend to choose a shorter path on move [34], each path ϕ's weight ω_ϕ among all possible paths is considered to be inversely proportional to $L(\phi)$ and computed as $\omega_\phi = L(\phi)^{-1} / \sum_{\phi \in \Phi'} (L(\phi)^{-1})$.
3) For each pair of directly connected regions $\langle r_k, r_{k+1} \rangle$ in path ϕ, its score is incremented by ϕ's weighted score so far. Formally, $score(\langle r_k, r_{k+1} \rangle) \mathrel{+}= PT.count * \omega_\phi$, where $PT.count$ is PT's count number in $\mathcal{H}_{PT}$.

[6]We omit the start and end regions $r_s^{\shortparallel}$ and $r_e^{\shortparallel}$ when the context is clear.

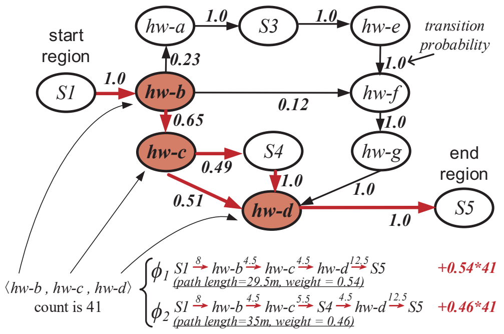
4) After each PT has been processed, we compute the **transition probability** from r_i to r_j as

$$\mathrm{P}_t(r_i, r_j) = \frac{score(\langle r_i, r_j \rangle)}{\sum_{r \in Out(r_i)} score(\langle r_i, r \rangle)} \tag{4}$$

where $Out(r_i)$ is the set of all directly connected regions in G_R that an object can enter after leaving r_i.

The mobility knowledge construction receives a full set of annotated ms-sequences, and returns a hash table that stores the candidate path set and transition probabilities for each stay region pair[7]. The detailed algorithm is given in Algorithm 6 in Appendix A.3.

Example 5. *Figure 8 gives an example of mobility knowledge construction with respect to Example 4. We organize all candidate paths in a sub-graph of G_R for the sake of presentation. For a region pattern $PT = \langle hw\text{-}b, hw\text{-}c, hw\text{-}d \rangle$, two candidate paths ϕ_1 and ϕ_2 that support PT are found, with respective weight* 0.54 *and* 0.46*. Subsequently, each pair of directly connected regions in ϕ_1 (e.g., $\langle S1, hw\text{-}b \rangle$ and $\langle hw\text{-}b, hw\text{-}c \rangle$) is added with $PT.count * 0.54 = 41 * 0.54$ and each pair in ϕ_2 is added with $41 * 0.46$. After all region patterns in $\mathcal{H}_{PT}$ have been processed, we compute the transition probability for every two directly connected regions. As indicated by the numbers (on the edges) in Figure 8, an object currently seen in hw-b has a probability 0.65 to enter hw-c and only a probability 0.12 to enter hw-f.*


start region
S1
1.0
hw-a
0.23
hw-b
0.65
1.0
S3
1.0
hw-e
transition probability
1.0
hw-f
0.12
1.0
hw-g
hw-c
0.49
S4
0.51
1.0
hw-d
1.0
end region
S5
1.0
⟨hw-b , hw-c , hw-d⟩ count is 41
φ1 S1 →8 hw-b →4.5 hw-c →4.5 hw-d →12.5 S5
(path length=29.5m, weight = 0.54)
+0.54*41
φ2 S1 →8 hw-b →4.5 hw-c →5.5 S4 →4.5 hw-d →12.5 S5
(path length=35m, weight = 0.46)
+0.46*41
Fig. 8. Example of Mobility Knowledge Construction

Note that the mobility knowledge can be updated when batches of m-semantics from the annotation layer are available. We evaluate using such a continuous updating paradigm in Section 6.1.4.

5.2 Missing M-Semantics Inference

Given the incomplete observations in an ms-sequence, we infer its missing m-semantics in two steps, namely a *most-likely path inference* (Section 5.2.1) and a *time period inference* (Section 5.2.2). The algorithm is formalized as Algorithm 7 in Appendix A.4.

5.2.1 Most-likely Path Inference.

Without loss of generality, we consider a simple problem form: Given an observed ms-sequence $\Lambda_o^{(o)} = \langle \lambda_s^{\shortparallel}, \lambda_q^{\triangleright}, \lambda_e^{\shortparallel} \rangle$, our path inference aims to find the mostly-like path that supports its region pattern $PT^{(o)} = \langle r_s^{\shortparallel}, r_q^{\triangleright}, r_e^{\shortparallel} \rangle$.

Given the candidate path set Φ constructed for $\langle r_s^{\shortparallel}, r_e^{\shortparallel} \rangle$, each path $\phi \in P$ that supports $PT^{(o)}$ can be denoted as $r_s^{\shortparallel} \to r_a^{\triangleright} \to \ldots \to r_b^{\triangleright} \to r_q^{\triangleright} \to r_c^{\triangleright} \to \ldots \to r_d^{\triangleright} \to r_e^{\shortparallel}$, where $r_a^{\triangleright} \to \ldots \to r_b^{\triangleright}$ and $r_c^{\triangleright} \to \ldots \to r_d^{\triangleright}$ are the *missing sub-paths* between two consecutive observed regions in $PT^{(o)}$. Note that two consecutive regions in $PT^{(o)}$ are usually not directly connected since the raw indoor positioning records are usually discrete.

[7]In fact, mobility knowledge is only constructed for a small fraction of region pairs, as we set an upper limit to the path length threshold γ to constrain the candidate path generation.

Following the literature [30, 37] in human mobility predicition, we assume the object movement between regions is a first-order Markov stochastic process, i.e., a region where an object is currently in is only related to the previous region it went through. Given an observed pattern $PT^{(o)}$, a path ϕ's *posterior probability* $\mathsf{P}(\phi|PT^{(o)})$[8] satisfies the expression below.

$$\mathsf{P}(\phi|PT^{(o)}) \;\propto\; \mathsf{P}(r_q^{\triangleright}|r_b^{\triangleright})\prod_{x=a}^{b-1}\mathsf{P}(r_{x+1}^{\triangleright}|r_x^{\triangleright})\mathsf{P}(r_a^{\triangleright}|r_s^{\shortparallel})\cdot\mathsf{P}(r_e^{\shortparallel}|r_d^{\triangleright})\prod_{y=c}^{d-1}\mathsf{P}(r_{y+1}^{\triangleright}|r_y^{\triangleright})\mathsf{P}(r_c^{\triangleright}|r_q^{\triangleright}) \tag{5}$$

where $\mathsf{P}(r_{x+1}^{\triangleright}|r_x^{\triangleright})$ is equivalent to our captured transition probability $\mathsf{P}_t(r_x^{\triangleright}, r_{x+1}^{\triangleright})$ in Equation 4. To find a most-likely path, we formulate it as a *maximum a posteriori* problem:

$$\begin{aligned}\arg\max_{\phi}\mathsf{P}(\phi|PT^{(o)}) = &\arg\max_{r_a^{\triangleright}\to\ldots\to r_b^{\triangleright}\subseteq\phi}\;\mathsf{P}_t(r_s^{\shortparallel}, r_a^{\triangleright})\prod_{x=a}^{b-1}\mathsf{P}_t(r_x^{\triangleright}, r_{x+1}^{\triangleright})\;\mathsf{P}_t(r_b^{\triangleright}, r_q^{\triangleright})\\ &\arg\max_{r_c^{\triangleright}\to\ldots\to r_d^{\triangleright}\subseteq\phi}\;\mathsf{P}_t(r_q^{\triangleright}, r_c^{\triangleright})\prod_{y=c}^{d-1}\mathsf{P}_t(r_y^{\triangleright}, r_{y+1}^{\triangleright})\;\mathsf{P}_t(r_d^{\triangleright}, r_e^{\shortparallel})\end{aligned} \tag{6}$$

The *max-product* algorithm [15] is used to solve this problem. By taking the transition probabilities from mobility knowledge as input, it finds an *optimal* sub-path between two consecutive observed regions in the observation $PT^{(o)}$ (e.g., $r_s^{\shortparallel}, r_q^{\triangleright}$ or $r_q^{\triangleright}, r_e^{\shortparallel}$) as the one with the maximum probability. By assembling the optimal sub-paths it finds, we can obtain a most-likely path for $\Lambda_o^{(o)}$.

If no candidate path ϕ is found to support the region pattern $PT^{(o)}$, we attribute it to the random errors that jump to another partition. In such a case, we modify $PT^{(o)}$ as follows. For any region r' in $PT^{(o)}$ that is not contained by any path in the candidate path set Φ, we change it to the most adjacent region r'' from those contained by any path in Φ.

Example 6. *Referring to the ms-sequence in Example 3, we obtain its observed region pattern* $PT^{(o)} = \langle S1, hw\text{-}b, hw\text{-}d, S5\rangle$. *To infer the most-likely path, we use the mobility knowledge to find the optimal sub-path between every two consecutive regions in* $PT^{(o)}$ *(c.f. Equation 6). Suppose that the optimal sub-paths between S1 and hw-b, hw-b and hw-d, hw-d and S5 are found as S1 ⟶ hw-b, hw-b ⟶ hw-c ⟶ hw-d, and hw-d ⟶ S5, respectively. The most-likely path is then assembled as S1 ⟶ hw-b ⟶ hw-c ⟶ hw-d ⟶ S5.*

5.2.2 Time Period Inference. The most-likely path inference finds an optimal sub-path for every two consecutive m-semantics in an observed ms-sequence. Next, we make each region $r_x \in \phi^*$ a temporal annotation to form the missing m-semantics between λ_p and λ_q.

For each such region r_x, its temporal annotation should be divided from the time period between λ_p and λ_q, i.e., $\mathsf{MT}_{p,q} = [\lambda_p.\tau.t_e, \lambda_q.\tau.t_s]$. However, it is hard to determine the time period that the object is in r_x as the object movement is unobserved during $\mathsf{MT}_{p,q}$. Also, the temporal annotations already made in other observed ms-sequences can hardly be used to infer the time period for r_x as the walking speed is variable and different across different objects. To ease computation, we assume that the moving object moves along its path with constant speed during $\mathsf{MT}_{p,q}$. Consequently, we can use the GRD between two regions as the reference[9] to divide r_x's time period from $\mathsf{MT}_{p,q}$. Formally, for a region r_x in $\phi^* = r_p \longrightarrow \ldots \longrightarrow r_q$, its time period is inferred as $\tau_x = [t_s^{(x)}, t_e^{(x)}]$ where

$$t_s^{(x)} = \lambda_p.\tau.t_e + \Delta t\cdot\frac{\sum_{i=p}^{x} dist_{gr}(r_p, r_i)}{\sum_{i=p}^{q} dist_{gr}(r_p, r_i)};\quad t_e^{(x)} = \lambda_p.\tau.t_e + \Delta t\cdot\frac{\sum_{i=p}^{x+1} dist_{gr}(r_p, r_i)}{\sum_{i=p}^{q} dist_{gr}(r_p, r_i)} \tag{7}$$

[8] Detailed derivation is given in Appendix B.2.
[9] As discussed in Section 4.2.2, a larger region costs more time to go out as indicated by its GRDs to its connected regions.

and $\Delta t = \lambda_q.\tau.t_s - \lambda_p.\tau.t_e$ in Equation 7 is the duration of $\mathrm{MT}_{p,q}$.

When the time period τ_x is inferred for r_x, we differentiate two cases. If r_x has already appeared in an observed m-semantics, the time period is added to the corresponding m-semantics. Otherwise, we should generate a missing m-semantics as $(r_x, \tau_x, pass)$, which means the object has passed through region r_x within the time period τ_x. We add each generated m-semantics to the observed ms-sequence and finish the complementing.

Example 7. *Continuing with Example 6, we process one of the optimal sub-paths hw-b → hw-c → hw-d. For the regions hw-b, hw-c and hw-d it contains, we divide the time period 9:16am-9:19am into three slices according to Equation 7, i.e., 9:16am-9:18am, 9:18am-9:19am, and 9:19am-9:19am. As hw-b and hw-d have appeared in the observed ms-sequence, their time periods are added to the corresponding m-semantics. Moreover, we generate a missing m-semantics (hw-c, 9:18-9:19am, pass). Likewise, we process other sub-paths and obtain a complete ms-sequence:*

(S1, 9:05am-9:16am, stay) → (hw-b, 9:16am-9:18am, pass) → (hw-c, 9:18am-9:19am, pass) → (hw-d, 9:19am-9:20am, pass) → (S5, 9:20am-9:42am, stay)

6 EXPERIMENTAL STUDIES

All programs are in Java and run on an Intel Xeon E5-2660 2.20GHz machine with 8GB memory.

6.1 Experiments on Real Data

Setting. We collected the real data from a Wi-Fi based positioning system in a 7-floor shopping mall in Hangzhou, China from Jan 1 to Jan 31, 2017. The daily numbers of objects (i.e., device MAC addresses) and positioning records in the operating hours (10AM – 10PM) were around 7,647 and 2,907,904, respectively. As a result, we obtained a total of 237,057 p-sequences. According to our survey, the positioning data error based on MIWD varied from 2 to 25 meters; the average sampling rate was around 1/18 Hz, i.e., a device can be observed about once every 18 seconds. As the indoor partitions divided by walls and doors may be non-convex or imbalanced (long in one dimension but short in the other), we employed an existing decomposition algorithm [43] to divide those irregular partitions into regular ones. As a result, we obtained 3,742 indoor partitions and 6,534 doors. In the mall, 202 shops were selected as semantic regions based on the application needs. The semantic region graph and its associated partition R-tree were kept in memory as they together are only 12.6 MB. The shortest indoor paths between doors were pre-computed to speed up computations on MIWD and GRD. Their maximum memory consumption was 990.8 MB.

We used the Event Editor module in TRIPS [24] to visually annotate m-semantics on the p-sequences as we were unable to know a device's exact whereabouts. Among the ms-sequences we annotated, 1,004 ms-sequences (including 17,322 m-semantics) obtained from Jan 1 were used to initialize the $\mathcal{E}$-function, and the other 9,687 ms-sequences (including 125,544 m-semantics) from Jan 2 to Jan 31 formed the ground truth for evaluation. The purpose of this setting is to study the case when the initial training data is insufficient. Particularly, an individual stay m-semantics or a sequence of pass m-semantics formed a snippet, and each snippet was extracted as a 28-dimensional feature vector for the model training. From Jan 2 to Jan 30, $\mathcal{E}$-function was continuously enhanced by a co-training paradigm [9] in which the most confident predicted snippets (with posterior probability ≥ 0.9 or ≤ 0.1) were added to the training set at the end of each day. The mobility knowledge was initially constructed for Jan 1 and also updated daily. The candidate path sets were generated for 10,682 directed region pairs, and the average path set size was 7.7. We kept the mobility knowledge in memory as it only needs around 36.1 MB.

Performance Metrics. In our framework implementation, we used one thread for each object. Therefore, we study the efficiency of our proposed techniques in terms of *average running time* for processing an individual object's data.

We use two metrics to measure construction effectiveness. On the one hand, we define *point-wise accuracy* (PA) as the fraction of positioning records that receive correct annotations. As spatial and event annotations are involved for each record, we implement *spatial PA* (SPA) and *event PA* (EPA) that measure the point-wise accuracy in terms of spatial annotations and event annotations, respectively. We then define *combined PA* (CPA) that linearly combines SPA and EPA by a tradeoff parameter α such that $CPA = \alpha \cdot SPA + (1 - \alpha) \cdot EPA$. Usually, the SPA requirement is stricter than EPA, as an event annotation makes no sense if the spatial annotation is wrong. On the contrary, analysts can still know a user's whereabouts provided the region is annotated correctly. Therefore, we use a large $\alpha = 0.7$ in the evaluation.

On the other hand, we note that sometimes the observed records are distributed unevenly, making some long-period m-semantics not get enough attention in accuracy evaluation. To this end, we segment each constructed ms-sequence at uniform intervals and measure the accuracy of each segment. The metric is called the *segment-wise accuracy* (SA). Similar to PA, we also implement the *spatial SA* (SSA), *event SA* (ESA), and *combined SA* (CSA). As the sampling rate of real data is relatively low, we set a segment length of 10 minutes. Note that if there is a cross of different m-semantics within a 10-min interval, we choose the one with a longer duration as the annotations of that segment. The process also applies to the ground truth. In general, PA can be used to assess the accuracy of annotations at a particular time, while SA measures the consistency between the annotation and ground truth from a global perspective.

6.1.1 Comparison of Annotation Methods. Our annotation method, denoted as *Dense-$\mathcal{E}$+LEC*, was implemented as two key modules. One is the temporal-event annotator depending on the density based splitting method (Section 4.1) and $\mathcal{E}$-function (Section 4.2.1). The other is the spatial annotator that mainly uses the location estimate confidence to match semantic regions for stay m-semantics (see Section 4.2.2). We designed several alternatives to compare with Dense-$\mathcal{E}$+LEC by modifying its modules. On the one hand, we implemented the SMoT method [4] (described in Section 4.1) for the temporal-event annotator. On the other hand, we used two different matching methods in the counterpart of the spatial annotator. The first called CTRD computes the centroid of all location estimates and selects the covering region as the spatial annotation. The second called VOTE counts the location estimates falling in each region and matches the one with the highest count. By combining these modifications, we obtained five alternative methods, i.e., *Dense-$\mathcal{E}$+CTRD*, *Dense-$\mathcal{E}$+VOTE*, *SMoT+LEC*, *SMoT+CTRD* and *SMoT+VOTE*. All alternatives are equipped with a cleaning layer and a complementing layer. Their performances are reported in Figure 9.

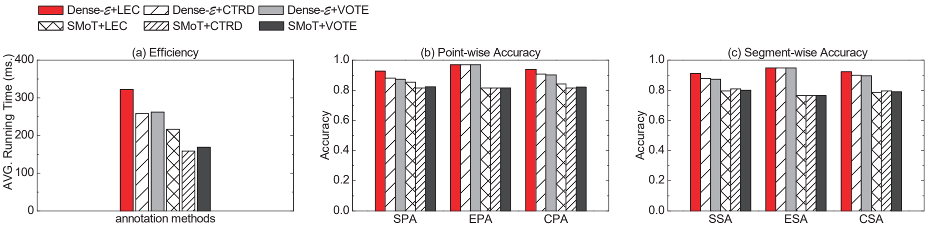


Fig. 9. Performance of Annotation Methods on Real Data

Referring to Figure 9(a), all methods based on Dense-$\mathcal{E}$ require more time to annotate m-semantics than those with SMoT. Despite the minor time spent in event identification, the time complexity is $O(n \cdot \log n)$ for the density based splitting and only $O(n)$ for SMoT, where n is the record number of a p-sequence. Comparing the three methods that use the same temporal-event annotator (i.e., Dense-$\mathcal{E}$ or SMoT), the method using LEC costs the most in making spatial annotations as it has to evaluate the confidence for each location estimate. The cost is slightly higher if a method uses VOTE other than CTRD, as VOTE needs to rank all involved regions and select the best. Nevertheless, our Dense-$\mathcal{E}$+LEC method can process a p-sequence within 350 milliseconds, which is very efficient for most analysis applications.

Figure 9(b) reports the three PAs for all annotation methods. The methods with Dense-$\mathcal{E}$ have the same EPA of 0.968, which outperforms 0.816 for those with SMoT. This shows that the spatiotemporal density information is more useful than the speed information in splitting p-sequences and identifying mobility events. Looking at the SPA, the spatial matching method LEC always outperforms VOTE and CTRD when combining with either Dense-$\mathcal{E}$ or SMoT. The CPA performs the best at 0.9264 with the method Dense-$\mathcal{E}$+LEC. The SAs reported in Figure 9(c) exhibit similar trends with the PAs but are lower in the testing. When a large number of records are observed in a short period, the PA tends to give a higher accuracy as each matched record count one for the hits. In contrast, the SA is only processed for each 10-min segment and thus is less affected by such uneven data distribution. Similarly, Dense-$\mathcal{E}$+LEC performs the best with the highest SSA, ESA, and CSA at 0.912, 0.949, and 0.923, respectively.

To sum up, our proposed method Dense-$\mathcal{E}$+LEC achieves a very good balance between efficiency and effectiveness, and therefore it is very useful in annotating m-semantics.

6.1.2 Effect of Cleaning and Complementing. We compared our complete three-layer framework *IMS-CA$\underline{C}$* (C denotes cleaning and $\underline{C}$ denotes complementing) with several alternatives. Specifically, *IMS-A* only contains a single annotation layer, two-layer method *IMS-CA* uses a cleaning layer before annotation, whereas another two-layer *IMS-A$\underline{C}$* adds a complementing layer after the annotation layer. Note that the annotation layer is necessary for constructing m-semantics. We report the construction results on the number of m-semantics per ms-sequence, PA and SA for these four methods in Table 3.

Table 3. Effect of Cleaning and Complementing on Real Data

Method	**# of M-Semantics**	SPA	EPA	CPA	SSA	ESA	CSA
IMS-A	11.94	0.8682	0.9005	0.8779	0.3941	0.4333	0.4059
IMS-CA	10.23	**0.9390**	**0.9684**	**0.9264**	0.4883	0.4543	0.4781
IMS-A$\underline{C}$	14.51	0.8682	0.9005	0.8779	0.7459	0.7539	0.7483
IMS-CA$\underline{C}$	14.12	**0.9390**	**0.9684**	**0.9264**	**0.9122**	**0.9487**	**0.9232**

IMS-A is the worst as it directly annotates m-semantics on the raw data. Its CPA and CSA are 0.8779 and 0.4059, respectively. Due to the low sampling frequency in the data, SA measures of those methods without data complementing (i.e., IMS-A and IMS-CA) are rather low. In comparison, IMS-CA with a cleaning layer significantly outperforms IMS-A; its CPA is increased at 0.9264 with improvement on both SPA and EPA measures, and all its SA measures also improve much. Correspondingly, the number of m-semantics per ms-sequence is decreased from 11.94 to 10.23. These results indicate that our cleaning method repairs many positioning data errors and removes their resultant wrong m-semantics. Therefore, the cleaning layer can improve the input data for subsequent layers. This effect of cleaning is also observed when we compare the PA of IMS-CA and IMS-A$\underline{C}$. Although the latter produces more m-semantics, many of them are problematic as they result from uncleaned data. Therefore, all PA measures of IMS-A$\underline{C}$ are considerably lower than their counterparts of IMS-CA.

IMS-CA$\underline{C}$ is always the best among all. Both its CPA and CSA are greater than 0.92. Note that the complementing layer takes no effects on PA measures as the hits are counted only over those observed positioning records. Therefore, the PA measures of IMS-CA equal to those of IMS-CA$\underline{C}$. However, the SA gap between them is distinguishable. These results show that IMS-CA$\underline{C}$ can produce reliable m-semantics highly consistent with the ground truth. With the help of our inference based complementing, IMS-CA$\underline{C}$ can recover the missing m-semantics that IMS-CA is unable to produce. As a result, the number of m-semantics per ms-sequence is increased from 10.23 to 14.12. More importantly, because of the combined effects of cleaning and complementing, IMS-CA$\underline{C}$'s accuracies improve clearly compared to IMS-CA and the others. In summary, the m-semantics construction on real data remarkably benefits from using our complete framework.

We also measure the average running time of processing an object's relevant data for each layer. The time costs in cleaning, annotation and complementing layers are around 42.6ms, 321.8ms, and 11.8ms, respectively. As the cleaning and complementing layers incur relatively very low time cost while improving the construction effectiveness significantly, it is beneficial and necessary to include them in our framework.

6.1.3 Comparison of Complementing Methods. As existing data repairing techniques [14, 38] are used for data instances and cannot directly support the tuple-form m-semantics, we designed several alternatives to verify the effectiveness of our complementing method. Recall that we used the max-product inference (denoted as MAX-P) in Section 5.2.1 to decode the most-likely indoor path to recover the spatial annotations of the missing m-semantics. We listed several comparable strategies as follows. First, a MIN-L strategy searches for a path with the *shortest length* between the observed regions as the most-likely path. Different from searching for the optimal path with the shortest length (i.e., MIN-L) or the highest product of transition probabilities (i.e., MAX-P), we can also randomly sample a path among all candidate paths based on their possibilities. The possibility of each candidate path can be measured by either *the reciprocal of path length* or *the product of transition probabilities*. The two strategies are denoted as Random-L and Random-P, respectively. In general, MIN-L and Random-L consider only static space information while MAX-P and Random-P use the transition probability computed upon both space information and historical mobility data. Besides, Random-P and Random-L output stochastic results while the other two give deterministic results. Combining with the time period inference presented in Section 5.2.2, we obtained four complementing methods correspondingly.

With the same cleaning and annotation layers we proposed, we measured the four complementing methods' SAs in the m-semantics construction. Referring to the SSAs reported in Figure 10, MAX-P performs the best while the other deterministic method MIN-L is clearly. Surprisingly, the stochastic method Random-P also slightly outperforms MIN-L, showing that the transition probabilities are more powerful in capturing the similar movements between regions than the path lengths. Compared to the path length, our transition probability considers both space information and historical object movements. Besides, Random-L is the worst of all. On the other hand, the ESA changes very slightly as the path inference methods have an impact on the boundary values. In general, our complementing based on the captured mobility knowledge is effective for the missing data recovery. Note that the above methods all select the most-likely path from a candidate set found by the A* algorithm. In this sense, their running time costs are very close.

6.1.4 Effect of Daily Updating Paradigm. We also study the effect of using a daily updating paradigm in constructing m-semantics. In the daily updating case, m-semantics annotated from the previous day were accumulated for re-training $\mathcal{E}$-function and updating mobility knowledge. In the case without daily updating, $\mathcal{E}$-function and mobility knowledge were only built with the data obtained from Jan 1. We measured the CSA for the IMS-CA and IMS-CA$\underline{C}$ methods in each day from Jan 2 to Jan 31. The methods without daily updating are marked with '*w/o DU*'. Referring to Figure 11, the measures of the methods without daily updating fluctuate a lot and in general decrease as

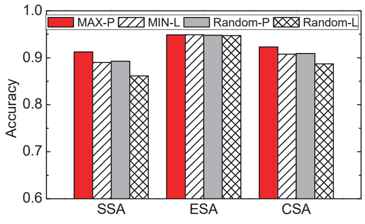


Fig. 10. Complementing Comparison

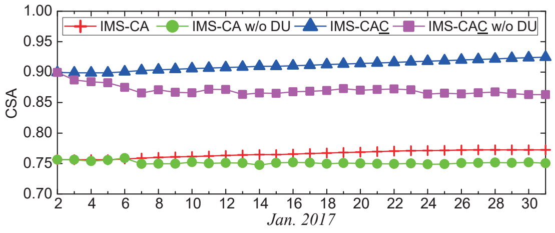


Fig. 11. Effectiveness vs. Daily Updating on Real Data

the day goes on, whereas those of the methods with daily updating improve and stabilize as time goes by. When the daily updating paradigm is employed, $\mathcal{E}$-function and mobility knowledge are continuously enhanced by more annotated m-semantics. Note that IMS-CA$\underline{C}$ increases more rapidly than IMS-CA as it exploits an additional complementing layer where the mobility knowledge can be periodically updated. The results show it is very helpful to update $\mathcal{E}$-function and mobility knowledge when raw positioning data is continuously streamed in.

6.1.5 M-Semantics' Ability to Answer Queries. We also evaluated our constructed m-semantics' ability to answer typical queries. Given a query set Q of indoor semantic regions and a query time interval QT, we introduce two top-k queries.

1) A *Top-k Popular Region Query* (TkPRQ) finds k semantic regions from Q that have the most number of *visits*[10] within QT.
2) A *Top-k Frequent Region Pair Query* (TkFRPQ) finds k most frequent pairs of semantic regions from $Q \times Q$ that both have been visited by the same object within QT.

TkPRQ and TkFRPQ are very useful in studies like popular indoor location discovery [25, 26] and frequent pattern mining [22]. Other than m-semantics constructed by the four methods in Table 3, we use the corresponding raw data and cleaned raw data to answer the two queries. The two corresponding methods are denoted as *RAW* and *RAW-C*, respectively. We introduce a naive strategy to each method, i.e., we compute the visits for all query regions (or region pairs) and return the top-k results by a full ranking. As no semantics is provided in the raw positioning data, the interpretation of *a visit to a semantic region* is done for the RAW and RAW-C methods. In particular, if an object's reported locations have been falling in a region over a time interval, the object is considered to have visited that region for once. We set the corresponding time interval to 1.5 min in RAW and 3 min in RAW-C for an optimized tuning.

We compared all methods' efficiency in terms of query execution time. Besides, we evaluated their effectiveness with respect to the ground truth results computed from the ground truth m-semantics described in the experimental setting. In particular, we used the metric *precision* that measures the ratio of the true top-k regions (or region pairs) in the returned top-k results. We issued 20 random queries for each query type and report the average efficiency and effectiveness measures. We fixed $k = 60$ and randomly pick 101 (50% of all) semantic regions to the query set Q. We varied the query time interval QT as 60, 120, 180, 240 min.

The efficiency results for TkPRQ and TkFRPQ are reported in Figure 12(a) and (b), respectively. In each test, the four IMS methods are faster than RAW and RAW-C by almost two orders of magnitude. When QT increases, more data (positioning records in RAW/RAW-C and m-semantics in IMS methods) should be loaded, and both queries incur more time to return the results. As the scale of raw data is much greater than that of m-semantics, RAW's and RAW-C's execution time increases more rapidly than all IMS methods. In fact, raw positioning data collected in a month was around 3.44 GB, whereas m-semantics constructed by IMS-CA$\underline{C}$ was only 220.1 MB. Moreover,

[10]In the query context, a visit is equivalent to a stay event.

IMS-CA$\underline{C}$ can return the results for both T*k*PRQ and T*k*FRPQ within one second for a four-hour query. These results verify that our constructed m-semantics are very efficient in answering the two top-k queries.

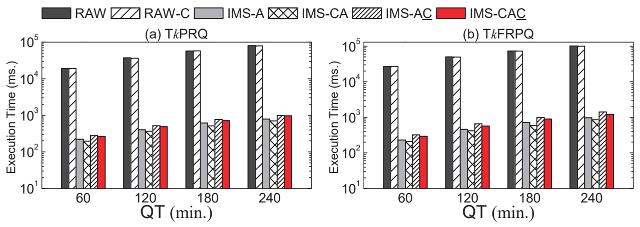


Fig. 12. Query Answering Efficiency vs. QT on Real Data

We also measured the effectiveness in each aforementioned setting. As shown in Figure 13(a) and (b), for both types of queries, the precision of all methods decreases with an increasing QT. When a longer QT is used, more relevant data should be considered in the query processing, which involves more data errors and makes the results less effective. Nevertheless, all methods with cleaning (i.e., RAW-C, IMS-CA, and IMS-CA$\underline{C}$) decrease very slowly. Among them, IMS-CA and IMS-CA$\underline{C}$ always outperform RAW-C, showing that the brief information kept in m-semantics can capture the underlying object movements very well. Moreover, when QT increases to 240 min, m-semantics constructed by IMS-CA$\underline{C}$ can still achieve a precision 82.8% for T*k*PRQ and 79.1% for T*k*FRPQ.

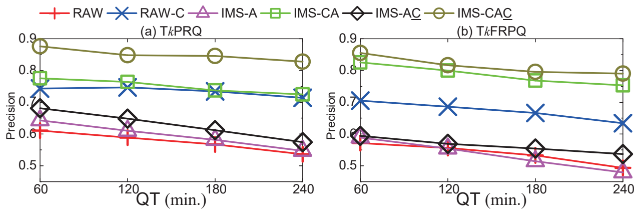


Fig. 13. Query Answering Effectiveness vs. QT on Real Data

We also studied the effectiveness by varying other parameters. In general, the precision results on varying k and $|Q|$ also verify that the m-semantics constructed by IMS-CA$\underline{C}$ are highly effective in answering both queries than other methods. Such additional results can be found in Appendix C.1.

6.2 Experiments on Synthetic Data

We used synthetic data to further verify our proposals' effectiveness when different levels of temporal sparsity and positioning errors are present in the raw indoor positioning data.

By using the indoor mobility data generator *Vita* [23], we simulated a 10-floor building environment with 4 staircases, 1,410 indoor partitions and 2,200 doors. A total of 423 semantic regions were decided upon the partitions at random. We generated moving objects in the environment for four hours. Specifically, 10K objects were distributed to the floors, each having a lifespan varied from ten seconds to four hours. Object maximum speed was set to $V_{max} = 1.7m/s$ and object movements followed the waypoint model [20]. In particular, each semantic region is considered as a destination, an object moves towards its destination along a pre-planned indoor path, it stays at the destination for a random period from 1 second to 30 min after arrival, and it moves again to the next destination that is decided randomly. We recorded an object's location every second as the ground truth trajectory, and generated its true m-semantics according to the simulated behavior, i.e., staying at (moving towards) a destination was regarded as a stay (pass) event. We obtained 998,618 ground truth m-semantics from the 10K objects' ms-sequences.

The synthetic IPT is maintained according to the ground truth trajectories as follows. After an object has sent an update to IPT, it keeps silent for at most T seconds. The *maximum positioning period* T refers to the maximum value of the time interval between two consecutive positioning records of an object. A location update is randomly within μ meters from the true location. False floor values and location outliers are added to the updates with certain probabilities (3% and 3%, respectively). In particular, a false floor value is produced within two floors up or down, and an outlier is randomly within 2.5μ-10μ meters from the true location. The *positioning error factor* μ measures the average distance between the positioning location and its true location. To test the effects of temporal sparsity and positioning error, we varied T and μ, respectively. The synthesized IPT instances as listed in Table 4.

Table 4. Synthetic IPT Instances

IPT Instance	Parameter Setting	# of Generated Records
$T5\mu3$	$T = 5s, \mu = 3m$	15,231,971
$T5\mu4$	$T = 5s, \mu = 4m$	15,230,508
$T5\mu5$	$T = 5s, \mu = 5m$	15,218,742
$T10\mu3$	$T = 10s, \mu = 3m$	7,416,906
$T15\mu3$	$T = 15s, \mu = 3m$	4,945,824

The memory consumptions for G_R, shortest indoor paths between doors, and mobility knowledge were 13.6 MB, 458 MB, and 48 MB, respectively. We randomly selected 3% of the ground truth ms-sequences to train $\mathcal{E}$-function, and the rest (including 968,660 m-semantics) were used as testing data in the evaluation.

M-Semantics Construction Effectiveness. We tested the four methods in Table 3 on different IPT instances. Here we use the SA with a segment length of 10 min to measure the consistency between constructed m-semantics and the ground truth described above. First, we fixed $\mu = 3m$ and vary T. The SSA and ESA are reported in Figure 14(a) and (b), respectively. When varying T from 5s to 15s, i.e., the observed data becomes sparser (see Table 4), and all methods' two SA measures decrease but IMS-CA$\underline{C}$'s decreases the slowest. Also, the performance gap between IMS-CA$\underline{C}$ and IMS-CA tends to expand when a larger T is involved, showing that our data complementing is very effective at recovering the missing m-semantics when the positioning data becomes sparser. When T=15s, IMS-CA$\underline{C}$ can still have an SSA of 0.91 and an ESA of 0.90. Still, IMS-A's accuracy is the worst and decreases rapidly when T increases. IMS-CA and IMS-CA$\underline{C}$ equipped with a cleaning layer clearly outperform the other two in all tests.

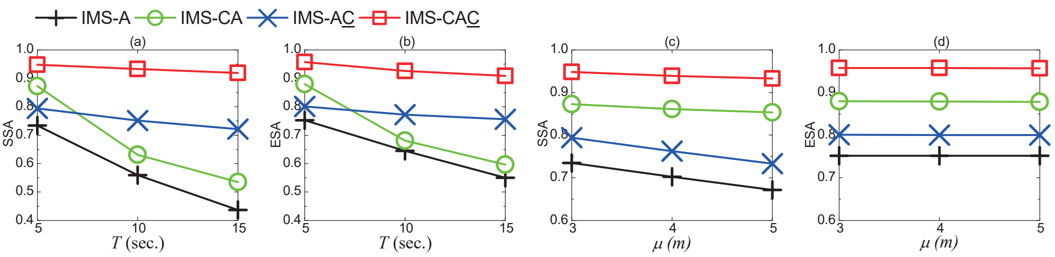


Fig. 14. Effectiveness of M-Semantics Construction on Synthetic Data

We also fixed T to 5s and tested with different μs. Refer to the two SA measures reported in Figure 14(c) and (d). When μ increases, the SSA of both IMS-CA$\underline{C}$ and IMS-CA stay stable while the others without cleaning decrease rapidly. This demonstrates that our raw data cleaning is very useful to reduce the negative impact of positioning errors. Besides, IMS-CA$\underline{C}$ outperforms IMS-CA in all tests due to the benefits of the complementing. On the other hand, the ESA of all methods is insensitive to μ and our method's ESA can still be higher than 0.95.

The results reported in different T and μ values show that the IMS-CA$\underline{C}$ framework works very effectively at constructing m-semantics even the raw data quality is relatively low.

With the ground truth trajectories in synthetic data, we also studied the effectiveness of our raw data cleaning method alone. The additional results in Appendix C.2 demonstrate that our raw data cleaning method is still effective for different T and μ values.

M-Semantics' Ability to Answer Queries. For each IPT instance, we answered the TkPRQ and TkFRPQ queries with the six methods introduced in Section 6.1.5. In the experiments, the time interval that indicates a visit to a region was tuned to 2 min in RAW and 3.5 min in RAW-C. A total of 212 (50% of all) semantic regions were picked to form query set Q, k was set to 60, and QT to 120 min. The precisions of different methods are reported in Figure 15.

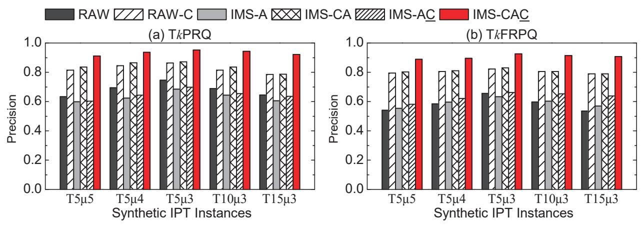


Fig. 15. Query Answering Effectiveness on Synthetic Data

Referring to the results of $T5\mu5$, $T5\mu4$ and $T5\mu3$ in Figure 15(a) and (b), precisions of all methods decrease when μ increases in both queries. All methods with data cleaning outperform the others in the tests. When μ increases to $5m$, our proposed IMS-CA$\underline{C}$ can still have a precision of 91.06% for TkPRQ and 88.84% for TkFRPQ, showing very high effectiveness in answering the two queries when the raw data contains many errors.

Referring to the results of $T5\mu3$, $T10\mu3$ and $T15\mu3$ for the two queries in Figure 15(a) and (b), the precision of each method decreases with an increasing T. However, IMS-CA$\underline{C}$ using the complementing decreases very slightly, while RAW-C and IMS-CA deteriorate more rapidly. These results show that our complementing is very useful to improve the constructed m-semantics, especially when the raw data is temporally sparse. IMS-A performs very poorly for both queries—sometimes it is even worse than directly using the raw data. Thus, it is very necessary to employ all three layers in constructing reliable m-semantics for query use.

7 RELATED WORK

Semantic Trajectory Representation. Parent et al. [33] propose the concept of semantic trajectory as a (GPS) data trace enhanced with annotations and/or complementary segmentations. Güting et al. [16] generalize this concept to symbolic trajectory, a sequence of pairs of a time interval and a label that refers to some particular user semantics. Marketos et al. [31] design a trajectory reconstruction method to transform raw trajectories into key movement features needed by specific warehousing applications. Zheng et al. [47] describe a trajectory only by some stay points where the moving objects stop for a relatively long time. Su et al. [39] propose a partition-and-summarization approach, in which a raw trajectory is segmented according to moving object's behavior, and the characteristics of each trajectory segment are summarized by a short text. Nogueira et al. [32] propose a framework with ontology to enrich GPS traces with Linked Open Data (LOD). Compared to these works, the m-semantics proposed in this study provide a structured representation about where-when-what of indoor users, which facilitates mobility analytics applications like semantic location prediction [37] and activity recommendation [47].

Semantic Annotation. Zhang et al. [46] derive fined-grained sequential patterns by a top-down splitting of the coarse-grained patterns obtained from POI grouping. Alvares et al. [4] extract stop and move events from trajectory points based on geographical information. Cao et al. [10] propose techniques for extracting from GPS data semantically meaningful geographical locations visited by users. Teng et al. [36] identify indoor stop-by pattern as a sequence of occurrence regions from uncertain RFID data. Different from the sequential patterns [46] and visiting location patterns [4, 10, 36], our work annotates two generic mobility patterns stay and pass, which are flexible for analyzing user behaviors by combining relevant information from indoor regions.

Liao et al. [27] extract activity types and significant places from a person's GPS traces using a hierarchical CRF. Yan et al. [44] propose an HMM based annotation method to infer stops and POI category for raw GPS records. Wu et al. [42] annotate location records with keywords extracted from geo-referenced social media data by kernel density estimation. By analyzing spatiotemporal regularity, Wu et al. [41] study the personalized annotation that enriches personal GPS records with the POI category. Our work differs from these works in several aspects. First, our work searches for a particular region as spatial annotation while other works focus on inferring textual [42] or categorical [27, 41, 44] information for location records. Second, our work uses the spatiotemporal characteristics of a positioning sequence under indoor topology to make annotations, while other works demand additional knowledge in geographic spaces, e.g., human activity regularity [27, 41] and POI category [44]. However, such priors are difficult to meet in indoor spaces with relatively small extents but complex topology. Third, our work makes use of historical mobility knowledge to recover the missing annotations, while none of these studies consider the data complementing after annotation. Unfortunately, the data sparsity in a compact indoor space poses even greater challenges than the wider-range geographical space.

Mobility Data Cleaning and Repairing. Mobility data can be in the form of a sequence of geometric locations (e.g., GPS or Wi-Fi positioning data) or symbolic locations (e.g., RFID reader or Bluetooth hotspot). Smoothing based methods [14, 38] are commonly used for handling noisy geometric locations by mathematically interpolating the points on the time series. In contrast, constraint based methods use additional information such as route structure [35] or other sensory data [12] to mitigate the positioning error. Differently, our cleaning method utilizes the indoor topology that is neither embedded in the smoothing based methods [14, 38] nor considered by the existing constraint based methods [12, 35]. The interpolation methods [14, 38] require inputs captured as points, and thus such methods fall short in complementing m-semantics for which the inputs are regions and three-tuple events.

Similar to the most-likely path inference in our m-semantics complementing, Zheng et al. [45] and Wei et al. [40] study the possible route inference for road networks and free spaces, respectively. However, these techniques are different from our proposal. First, the routable graph built in [40] regards that each two adjacent geospatial grids are connected, unable to model the topological constraints dominated in indoor spaces. Second, the graph based inference and the nearest neighbor based inference in [45] models transition probabilities using only the frequency of relevant historical trajectories. In contrast, region transition probabilities defined in our study considers both movement pattern frequency and distance weights between regions. In Section 6.1.3, we have demonstrated that the mobility knowledge that combines both movement and distance information significantly enhances our path inference.

Using indoor topology to clean and/or repair symbolic location sequences has been studied. Chen et al. [11] clean RFID readings by a Bayesian inference approach that makes use of duplicate RFID readings and prior knowledge about the environment and readers. Baba et al. clean RFID tracking data by utilizing either the integrity constraints [6, 7] implied by RFID reader deployment or the relevant knowledge [5] learned from historical data. In their early solutions, distance-aware graph [7] and probabilistic graph [6] are designed to handle the false positives and false negatives, respectively. Their learning based approach [5] uses an indoor RFID Multi-variate HMM to build the correlation of indoor object locations and RFID readings to handle both false positives and false negatives. Compared to the RFID data captured as discrete states at different timestamps, the Wi-Fi positioning locations in our setting are continuous coordinates that can hardly be optimized with unlimited possible values. Also, our cleaning method identifies and repairs three types of data errors in one pass, while works [6, 7] can only handle one specific type of data error in a separate and specialized process. Moreover, our inference method for recovering missing data is different from the state detection model [11], probabilistic graph [6], and multi-variate HMM [5] in three aspects. First, our inference is based on the mobility constraints captured at the m-semantics level. Second, we model the similar movements between each pair of semantic regions, while works [5, 6, 11] build prior knowledge in a global scope. Third, unlike other works, our transition probability computation considers the effect of the corresponding path's walking length.

8 CONCLUSION AND FUTURE WORK

This paper tackles the problem of translating mobility semantics from raw indoor positioning data. We propose a three-layer framework with a set of novel techniques. In the cleaning layer, we design a mobility constraint based cleaning method that eliminates indoor positioning data errors. In the annotation layer, we design a density based method to split the cleaned data sequence into snippets according to the spatiotemporal densities of the data, and a semantic matching method to make proper annotations and decide mobility semantics for the snippets. In the complementing layer, we devise an inference method to recover the missing mobility semantics with the mobility knowledge captured from historical data. The experiments on real and synthetic data verify that our framework is efficient and effective in constructing mobility semantics, and the constructed mobility semantics are able to answer typical queries efficiently and effectively.

For future work, it is useful to incorporate the temporal annotations of mobility semantics in inferring the missing data. It is also interesting to model and learn the probabilistic dependencies among the attributes of mobility semantics, in order to annotate the most-likely semantic regions and mobility events for the positioning sequences. In addition, it is useful to enrich the mobility semantics by making use of other user behavior data such as transaction logs or check-ins.

ACKNOWLEDGMENTS

This work was supported by the Independent Research Fund Denmark (No. 8022-00366B), the National Science Foundation of China (Nos. 61672455 and 61528207), and the Natural Science Foundation of Zhejiang Province of China (No. LY18F020005).

REFERENCES

[1] Industrial IoT World. https://iiot-world.com/connected-industry/top-industries-using-indoor-positioning/.
[2] Smartphone penetration. https://en.wikipedia.org/wiki/List_of_countries_by_smartphone_penetration#2018_rankings.
[3] Smile - Statistical Machine Intelligence and Learning Engine. http://haifengl.github.io/smile/.
[4] L. O. Alvares, V. Bogorny, B. Kuijpers, J. de Macedo, B. Moelans, and A. Vaisman. A model for enriching trajectories with semantic geographical information. *Advances in geographic information systems*, 22, 2007.
[5] A. I. Baba, M. Jaeger, H. Lu, T. B. Pedersen, W.-S. Ku, and X. Xie. Learning-based cleaning for indoor RFID data. In *SIGMOD*, pp. 925–936, 2016.
[6] A. I. Baba, H. Lu, T. B. Pedersen, and X. Xie. Handling false negatives in indoor RFID data. In *MDM*, pp. 117–126, 2014.
[7] A. I. Baba, H. Lu, X. Xie, and T. B. Pedersen. Spatiotemporal data cleansing for indoor RFID tracking data. In *MDM*, pp. 187–196, 2013.
[8] D. Birant and A. Kut. ST-DBSCAN: An algorithm for clustering spatial–temporal data. *Data & Knowledge Engineering*, 60(1): 208–221, 2007.
[9] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, pp. 92–100, 1998.
[10] X. Cao, G. Cong, and C. S. Jensen. Mining significant semantic locations from GPS data. *PVLDB*, 3(1): 1009–1020, 2010.

[11] H. Chen, W.-S. Ku, H. Wang, and M.-T. Sun. Leveraging spatio-temporal redundancy for RFID data cleaning. In *SIGMOD*, pp. 51–62, 2010.
[12] J. Chen and M. Bierlaire. Probabilistic multimodal map matching with rich smartphone data. *Journal of Intelligent Transportation Systems*, 19(2): 134–148, 2015.
[13] B. Fang, S. Liao, K. Xu, H. Cheng, C. Zhu, and H. Chen. A novel mobile recommender system for indoor shopping. *Expert Systems with Applications*, 39(15): 11992–12000, 2012.
[14] V. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello. Bayesian filtering for location estimation. *Pervasive Computing*, 2(3): 24–33, 2003.
[15] A. Globerson and T. S. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *NIPS*, pp. 553–560, 2008.
[16] R H. Güting, F. Valdés, and M L. Damiani. Symbolic trajectories. *ACM Transactions on Spatial Algorithms and Systems*, 1(2): 7, 2015.
[17] V. Honkavirta, T. Perala, S. Ali-Loytty, and R. Piché. A comparative survey of WLAN location fingerprinting methods. In *WPNC*, pp. 243–251, 2009.
[18] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8): 57–66, 2001.
[19] P. L. Jenkins, T. J. Phillips, E. J. Mulberg, and S. P. Hui. Activity patterns of californians: use of and proximity to indoor pollutant sources. *Atmospheric Environment*, 26(12): 2141–2148, 1992.
[20] D. B. Johnson and D.A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pp. 153–181. 1996.
[21] N. Klepeis, W. Nelson, W. Ott, J. Robinson, A. Tsang, P. Switzer, J. Behar, S. Hern, and W. Engelmann. The National Human Activity Pattern Survey (NHAPS): a resource for assessing exposure to environmental pollutants. *Journal of Exposure Science*, 11(3): 231, 2001.
[22] M. B. Kjærgaard, M Wirz, D. Roggen, and G. Tröster. Mobile sensing of pedestrian flocks in indoor environments using WIFI signals. In *PerCom*, pp. 95–10, 2012.
[23] H. Li, H. Lu, X, Chen, G. Chen, K. Chen, and L. Shou. Vita: A versatile toolkit for generating indoor mobility data for real-world buildings. *PVLDB*, 9(13): 1453–1456, 2016.
[24] H. Li, H. Lu, F. Shi, G. Chen, K. Chen, and L. Shou. TRIPS: A system for translating raw indoor positioning data into visual mobility semantics. *PVLDB*, 11(12): 1918–1921, 2018.
[25] H. Li, H. Lu, L. Shou, G. Chen, and K. Chen. Finding most popular indoor semantic locations using uncertain mobility data. *IEEE Transactions on Knowledge and Data Engineering*, 31(11): 2108–2123, 2018.
[26] H. Li, H. Lu, L. Shou, G. Chen, and K. Chen. In search of indoor dense regions: An approach using indoor positioning data. *IEEE Transactions on Knowledge and Data Engineering*, 30(8): 1481–1495, 2018.
[27] L. Liao, D. Fox, and H. Kautz. Location-based activity recognition. In *NIPS*, pp. 787–794, 2006.
[28] H. Lu, X. Cao, and C. S. Jensen. A foundation for efficient indoor distance-aware query processing. In *ICDE*, pp. 438–449, 2012.
[29] H. Lu, C. Guo, B. Yang, and C. S. Jensen. Finding frequently visited indoor pois using symbolic indoor tracking data. In *EDBT*, pp. 449–460, 2016.
[30] X. Lu, E. Wetter, N. Bharti, A. J. Tatem, and L. Bengtsson. Approaching the limit of predictability in human mobility. *Scientific reports*, 3(10), 2013.
[31] G. Marketos, E. Frentzos, I. Ntoutsi, N. Pelekis, A. Raffaetà, and Y. Theodoridis. Building real-world trajectory warehouses. In *MobiDE*, pp. 8–15, 2008.
[32] T.P. Nogueira, R.B. Braga, C.T. de Oliveira, and H. Martin. FrameSTEP: A framework for annotating semantic trajectories based on episodes. *Expert Systems with Applications*, 92: 533–545, 2018.
[33] C. Parent, S. Spaccapietra, *et al.* Semantic trajectories modeling and analysis. *ACM Computing Surveys*, 45(4), 2013.
[34] T S. Prentow, H. Blunck, K. Grønbæk, and M. B. Kjærgaard. Estimating common pedestrian routes through indoor path networks using position traces. In *MDM*, pp. 43-48, 2014.
[35] T S. Prentow, A. Thom, H. Blunck, and J. Vahrenhold. Making sense of trajectory data in indoor spaces. In *MDM*, pp. 116-121, 2015.
[36] S-Y. Teng, W-S. Ku, and K-T. Chuang. Toward mining stop-by behaviors in indoor space. *ACM Transactions on Spatial Algorithms and Systems*, 3(2), 2017.
[37] S. Scellato, M Musolesi, C. Mascolo, V. Latora, and A. T. Campbell. Nextplace: a spatio-temporal prediction framework for pervasive systems. In *ICPC*, pp. 152–169. 2011.
[38] N. Schuessler and K W. Axhausen. Processing raw data from global positioning systems without additional information. *Transportation Research Record*, 2105(1): 28–36. 2009.
[39] H. Su, K. Zheng, K. Zeng, J. Huang, S. Sadiq, N. J. Yuan, and X. Zhou. Making sense of trajectory data: A partition-and-summarization approach. In *ICDE*, pp. 963–974. 2015.
[40] L. Wei, Y. Zheng, and W. Peng. Constructing popular routes from uncertain trajectories. In *KDD*, pp. 195–203, 2012.
[41] F. Wu and Z. Li. Where did you go: Personalized annotation of mobility records. In *CIKM*, pp. 589–598, 2016.
[42] F. Wu, Z. Li, W-C. Lee, H. Wang, and Z. Huang. Semantic annotation of mobility data using social media. In *WWW*, pp. 1253–1263, 2015.
[43] X. Xie, H. Lu, and T. B. Pedersen. Efficient distance-aware query evaluation on indoor moving objects. In *ICDE*, pp. 434–445, 2013.

[44] Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, and K. Aberer. Semantic trajectories: Mobility data computation and annotation. *ACM Transactions on Intelligent Systems and Technology*, 4(3): 49, 2013.
[45] K. Zheng, Y. Zheng, X. Xie, and X. Zhou. Reducing uncertainty of low-sampling-rate trajectories. In *ICDE*, pp. 1144–1155, 2012.
[46] C. Zhang, J. Han, L. Shou, J. Lu, and T. La Porta. Splitter: Mining fine-grained sequential patterns in semantic trajectories. *PVLDB*, 7(9): 769–780, 2014.
[47] K. Zheng and X. Xie. Learning travel recommendations from user-generated GPS traces. *ACM Transactions on Intelligent Systems and Technology*, 2(1): 2, 2011.
[48] W. Zeng and R. L. Church. Finding shortest paths on real road networks: the case for A*. *International Journal of GIS*, 23(4): 531–543, 2009.

Appendix A ALGORITHMS

A.1 Density based Splitting Algorithm

Algorithm 4 divides a cleaned p-sequence Θ_o into a sequence of data snippets. It mainly consists of a procedure of density based clustering (lines 2–19) and a procedure of p-sequence partitioning (lines 20–33).

First, a time-ordered sequence $\mathcal{A}_{snpt}$ is initialized to hold the split snippets (line 1). Subsequently, the density based clustering is conducted on the positioning records in the p-sequence Θ_o (lines 2–19). In particular, the function *RetrieveNeighbors* (lines 5 and 17) issues a range query on Θ_o to find a set $\mathcal{N}$ of neighboring records, each having its spatial distance to θ smaller than ϵ_s and temporal distance to θ smaller than ϵ_t. Besides, the function *GetAdaptivePtm* (lines 6 and 18) computes an adaptive pt_m associated with the current record.

To be specific, *GetAdaptivePtm* works as follows (lines 35–38). It first retrieves a set $\mathcal{N}_t$ of records within the time window $[t - \epsilon_t, t + \epsilon_t]$ by a temporal range query on Θ_o (line 36). Note that $|\mathcal{N}_t|$ is proportional to the local sampling rate st_{local} such that $|\mathcal{N}_t| = 2 * \epsilon_t * st_{local}$. Afterwards, a round-down sigmoid function $\lfloor \frac{e^{|\mathcal{N}_t|-N}}{1+e^{|\mathcal{N}_t|-N}} * 2P + B \rfloor$ is used to obtain the adaptive pt_m (line 37). Particularly, N, P, B are three integer parameters, and pt_m increases monotonically with $|\mathcal{N}_t|$ from B to $2P + B$ and equals to $P + B$ when $|\mathcal{N}_t| = N$. The optimal values of parameters ϵ_s, ϵ_t, N, P, and B can be determined by a grid search. The adaptive function here can be replaced with another design, e.g., an one that considers both the sampling rate and potentially located region.

When the clustering is done, each a cluster is converted into a dense snippet *snpt* and added to $\mathcal{A}_{snpt}$ (lines 20–23). After that, the algorithm iterates through each consecutive dense snippets $\langle snpt_i, snpt_{i+1} \rangle$ in $\mathcal{A}_{snpt}$ (lines 24–33). It first checks if the two dense snippets $snpt_i$, $snpt_{i+1}$ can be merged in the sense of the merging conditions defined on Δ_t and Δ_s (line 25). If it is, they are removed from $\mathcal{A}_{snpt}$ (line 26), merged into one (line 27), and added back to $\mathcal{A}_{snpt}$ (line 28). Otherwise, it further checks if there exists any positioning record labeled with *noise* between the two snippets (lines 29–30). If any of such record can be found, a non-dense snippet is formed by retrieving all such records in-between, and then added to $\mathcal{A}_{snpt}$ (lines 31–33). At the end, the time-ordered sequence $\mathcal{A}_{snpt}$ is returned (line 34).

A.2 Semantic Matching Algorithm

Algorithm 5 uses the $\mathcal{E}$-function and the semantic region graph to translate a snippet Θ_o^* into a sequence Λ_o^* of m-semantics. At the beginning, a time-ordered sequence Λ_o^* is initialized to hold the matched m-semantics (line 1). If the snippet is classified as *stay* by the function $\mathcal{E}(\Theta_o^*)$ (line 2), it is matched to a stay m-semantics (lines 2–9). In particular, the event and temporal annotations are made at first (line 3). Subsequently, the algorithm iterates through each positioning records θ_i and computes its location estimate confidence $conf^{(i)}$ (lines 4–6). As a result, the stay position is

Algorithm 4: DensityBasedSplitting

```
Input: p-sequence $\Theta_o$, temporal distance threshold $\epsilon_t$, spatial distance threshold $\epsilon_s$, tolerate time span
       $\Delta_t$, tolerate spatial distance $\Delta_s$.
Output: sequence of split snippets $\mathcal{A}_{snpt}$.
time-ordered sequence $\mathcal{A}_{snpt} \longleftarrow \langle\rangle$
$cluster\_id \longleftarrow 0$
for each record $\theta \in \Theta_o$ do
    if $label(\theta)$ is $null$ then
        $\mathcal{N} \longleftarrow RetrieveNeighbors(\theta, \epsilon_t, \epsilon_s)$
        $pt_m \longleftarrow GetAdaptivePtm(\Theta_o, \theta.t, \epsilon_t)$
        if $|\mathcal{N}| \leq pt_m$ then
            $label(\theta) \longleftarrow noise$
        else
            $cluster\_id \longleftarrow cluster\_id + 1$
            $label(\theta) \longleftarrow cluster\_id$
            seed set $\mathcal{S} \longleftarrow \mathcal{N} \setminus \theta$
            for each record $\theta' \in \mathcal{S}$ do
                if $label(\theta')$ is $noise$ then $label(\theta') \longleftarrow cluster\_id$
                if $label(\theta')$ is $null$ then
                    $label(\theta') \longleftarrow cluster\_id$
                    $\mathcal{N}' \longleftarrow RetrieveNeighbors(\theta', \epsilon_t, \epsilon_s)$
                    $pt'_m \longleftarrow GetAdaptivePtm(\Theta_o, \theta'.t, \epsilon_t)$
                    if $|\mathcal{N}'| > pt'_m$ then $\mathcal{S} \longleftarrow \mathcal{S} \cup \mathcal{N}'$
for each cluster id $cluster\_id$ do
    time-ordered sequence $snpt \longleftarrow \langle\rangle$
    add all records labeled with $cluster\_id$ to $snpt$
    mark $snpt$ as $dense$; add $snpt$ to $\mathcal{A}_{snpt}$
for any consecutive $dense$ snippets $\langle snpt_i, snpt_{i+1}\rangle$ in $\mathcal{A}_{snpt}$ do
    if $CanBeMerged(snpt_i, snpt_{i+1}, \Delta_t, \Delta_s)$ then
        $\mathcal{A}_{snpt} \longleftarrow \mathcal{A}_{snpt} \setminus \langle snpt_i, snpt_{i+1}\rangle$
        $snpt' \longleftarrow snpt_i \cup snpt_{i+1}$
        mark $snpt'$ as $dense$; add $snpt$ to $\mathcal{A}_{snpt}$
    else
        if exists a record labeled with $noise$ between the snippets then
            time-ordered sequence $snpt \longleftarrow \langle\rangle$
            add all records labeled with $noise$ in-between to $snpt$
            mark $snpt$ as $non\_dense$; add $snpt$ to $\mathcal{A}_{snpt}$
return $\mathcal{A}_{snpt}$
Function $GetAdaptivePtm$ $(\Theta_o, t, \epsilon_t)$
    $\mathcal{N}_t \longleftarrow$ range query on $\Theta_o$ within $[t - \epsilon_t, t + \epsilon_t]$
    $pt_m \longleftarrow \lfloor \frac{e^{|\mathcal{N}_t|-N}}{1+e^{|\mathcal{N}_t|-N}} * 2P + B \rfloor$
    return $pt_m$
```

inferred as $\hat{l}$ (line 7) and the spatial annotation r is matched by searching on G_R (line 8). Afterwards, the matched m-semantics λ is added to Λ_o^* (line 9).

Otherwise, if the snippet is corresponding to a pass event, we conduct the matching as follows (lines 10–20). First, an event annotation *pass* is made for each m-semantics to be matched, and a variable λ' is used to control the merge of consecutive pass m-semantics (line 11). The algorithm then iterates through each positioning record θ_i (lines 12–20). Specifically, the temporal annotation is generated (line 13) and the spatial annotation is determined as the region that contains $\theta_i.l$ (line 14). If the current θ_i's spatial annotation is the same as $\lambda'.r$, they are merged together (lines 15–16). Otherwise, λ' can no longer be merged with any subsequent m-semantics and it is added to Λ_o^*, and the current matched annotations (r, τ, δ) is assigned to λ' (lines 17–19). Note that λ' should be added to Λ_o^* in the last loop of iteration (line 20). Finally, Λ_o^* is returned as a sequence of matched m-semantics (line 21).

Algorithm 5: SemanticMatching

```
Input: split snippet $\Theta_o^*$, event identification function $\mathcal{E}$, semantic region graph $G_R$.
Output: sequence of m-semantics $\Lambda_o^*$.
1  time-ordered sequence $\Lambda_o^* \longleftarrow \langle\rangle$
2  if $\mathcal{E}(\Theta_o^*) = stay$ then
3      $\delta \longleftarrow stay$; $\tau \longleftarrow [head(\Theta_o^*).t, tail(\Theta_o^*).t]$
4      for each positioning record $\theta_i$ in $\Theta_o^*$ do
5          $\mathcal{N}^{(i)} \longleftarrow$ find the $k$ nearest neighboring records of $\theta_i$
6          $conf^{(i)} \longleftarrow \left(\frac{\sum_{\theta_j \in \mathcal{N}^{(i)}} dist_I(\theta_i.l, \theta_j.l)}{|\mathcal{N}^{(i)}|}\right)^{-1}$
7      $sum\_conf \longleftarrow \sum_{\theta_i \in \Theta_o^*} conf^{(i)}$; $\hat{l} \longleftarrow \sum_{\theta_i \in \Theta_o^*} \frac{conf^{(i)}}{sum\_conf} \cdot \theta_i.l$
8      $r^* \longleftarrow$ search $G_R$ for a region that contains $\hat{l}$
9      $\lambda \longleftarrow (r^*, \tau, \delta)$; add $\lambda$ to $\Lambda_o^*$
10 else
11     $\delta \longleftarrow pass$; $\lambda' \longleftarrow null$
12     for each positioning record $\theta_i$ in $\Theta_o^*$ do
13         $\tau \longleftarrow [\theta_i.t, \theta_i.t]$
14         $r^* \longleftarrow$ search $G_R$ for a region that contains $\theta_i.l$
15         if $r^* = \lambda'.r$ then
16             $\lambda'.\tau \longleftarrow \lambda'.\tau \cup \tau$;
17         else
18             if $\lambda'$ is not $null$ then add $\lambda'$ to $\Lambda_o^*$
19             $\lambda' \longleftarrow (r^*, \tau, \delta)$
20         if $\theta_i = tail(\Theta_o^*)$ then add $\lambda'$ to $\Lambda_o^*$
21 return $\Lambda_o^*$
```

A.3 Mobility Knowledge Construction Algorithm

Algorithm 6 takes G_R and a set of ms-sequences as input and returns the mobility knowledge in a hash table $\mathcal{MK}$. At the beginning, the hash table $\mathcal{MK}$ is initialized to store the path set Φ as well as the corresponding transition probabilities *TP* for each *directed* pair of stay regions (line 1). For each such pair $\langle r_s^{\shortparallel}, r_e^{\shortparallel}\rangle$, a function *ConstructForOnePair* is called to obtain the corresponding Φ and *TP* (lines 2–3).

Function *ConstructForOnePair* works as follows (lines 5–24). First, it performs an *A*-Search* to find a set Φ of candidate paths (line 6) and constructs the transition probabilities *TP* for each two directly connected regions in the candidate paths (lines 7–23). In particular, a hash table $\mathcal{H}_{PT}$ that records the region patterns is constructed by processing each ms-sequence $\Lambda_o \in \mathcal{S}_\Lambda$ (lines 8–12). Afterwards, it records the score for each pair of directly connected regions in a candidate path such that a hash table $\mathcal{H}_s$ is constructed (lines 13–19). Consequently, the transition probabilities *TP* is computed with the scores recorded in $\mathcal{H}_s$ (lines 20–23). At the end, the candidate path set Φ is returned along with *TP* (line 24).

Algorithm 6: MobilityKnowledgeConstruction

```
   Input: semantic region graph $G_R$, set of original ms-sequences $\mathcal{S}_\Lambda$.
   Output: hash table $\mathcal{MK}$ to store the mobility knowledge for each pair of stay regions.
1  hash table $\mathcal{MK} : \langle R \times R \rangle \longrightarrow \langle \Phi, TP \rangle$
2  for each directed pair of stay regions $\langle r_s^{||}, r_e^{||} \rangle$ do
3      $\mathcal{MK}[\langle r_s^{||}, r_e^{||} \rangle] \longleftarrow ConstructForOnePair(G_R, \mathcal{S}_\Lambda, \langle r_s^{||}, r_e^{||} \rangle)$
4  return $\mathcal{MK}$
5  Function ConstructForOnePair ($G_R, \mathcal{S}_\Lambda, \langle r_s^{||}, r_e^{||} \rangle$)
6      candidate path set $\Phi \longleftarrow A^*\text{-}Search(G_R, r_s^{||}, r_e^{||})$
7      transition probability $TP : (R \times R) \longrightarrow$ probability
8      hash table $\mathcal{H}_{PT} : PT \longrightarrow$ count
9      for each ms-sequence $\Lambda_o$ in $\mathcal{S}_\Lambda$ do
10         for each matched segment $\langle \lambda_s^{||}, \lambda_i^{\triangleright}, \ldots, \lambda_j^{\triangleright}, \lambda_e^{||} \rangle$ in $\Lambda_o$ do
11             $PT \longleftarrow \langle r_i^{\triangleright}, \ldots, r_j^{\triangleright} \rangle$
12             $\mathcal{H}_{PT}[PT] \longleftarrow \mathcal{H}_{PT}[PT] + 1$
13     hash table $\mathcal{H}_s : (R \times R) \longrightarrow$ score
14     for each entry $\langle PT, count \rangle$ in $\mathcal{H}_{PT}$ do
15         $\Phi' \longleftarrow$ find a subset of paths that hold $PT$
16         for each path $\phi \in \Phi'$ do
17             $\omega_\phi \longleftarrow L(\phi)^{-1} / \sum_{\Phi'} L(\phi)^{-1}$
18             for each directly connected regions $\langle r_k, r_l \rangle$ in $\phi$ do
19                 $\mathcal{H}_s[\langle r_k, r_l \rangle] \longleftarrow \mathcal{H}_s[\langle r_k, r_l \rangle] + count * \omega_\phi$
20     for each region $r_i$ covered by path set $\Phi$ do
21         $Out(r_i) \longleftarrow$ find the enterable regions when leaving $r_i$
22         for each region $r_j$ in $Out(r_i)$ do
23             $TP[\langle r_i, r_j \rangle] \longleftarrow \frac{\mathcal{H}_s[\langle r_i, r_j \rangle]}{\sum_{r \in Out(r_i)} \mathcal{H}_s[\langle r_i, r \rangle]}$
24     return $\langle \Phi, TP \rangle$
```

A.4 M-semantics Inference Algorithm

Algorithm 7 gives the procedure of inferring the missing m-semantics for an original ms-sequence, by utilizing the mobility knowledge $\mathcal{MK}$ and the indoor mobility constraints captured in G_R. Given an observation $\langle \lambda_s^{||}, \ldots, \lambda_e^{||} \rangle$ between stay m-semantics $\lambda_s^{||}$ and $\lambda_e^{||}$ (line 1), the corresponding candidate path set Φ is loaded from $\mathcal{MK}[\langle r_s^{||}, r_e^{||} \rangle]$ (line 2), and a function *InferMostLikelyPath* is called to infer a most-likely path $\hat{\phi} \in \Phi$ for the observation between $\lambda_s^{||}$ and $\lambda_e^{||}$ (line 3). Furthermore, the algorithm makes use of the GRDs between directly connected regions to infer the time period for each region r_x contained in $\hat{\phi}$ (lines 4–10). In particular, a function *InferDurationTime* is called to infer the time period τ_x for each r_x (line 5). If the current r_x has been observed in an m-semantics λ_x in Λ_o (line 6), its inferred time period is merged with λ_x to form a new m-semantics, and the new m-semantics is added back to Λ_o (lines 7–8); otherwise, the algorithm maps r_x to a pass m-semantics and adds it to Λ_o (line 10). At the end, the complemented ms-sequence Λ_o is returned (line 11). The technical details of function *InferMostLikelyPath* and function *InferDurationTime* have been given in Section 5.2.1 and Section 5.2.2 of the paper, respectively.

Algorithm 7: MSemanticsInference

```
Input: an observed ms-sequence $\Lambda_o$, hash table $\mathcal{MK}$, semantic region graph $G_R$.
Output: a complemented ms-sequence $\Lambda_o$.
1  for each observation $\langle \lambda_s^{||}, \ldots, \lambda_e^{||} \rangle \subseteq \Lambda_o$ do
2      $\langle \Phi, TP \rangle \longleftarrow \mathcal{MK}[\langle r_s^{||}, r_e^{||} \rangle]$
3      $\hat{\phi} \longleftarrow$ InferMostLikelyPath($\Phi$, $TP$, $\langle \lambda_s^{||}, \ldots, \lambda_e^{||} \rangle$)
4      for each region $r_x$ in $\hat{\phi}$ do
5          $\tau_x \longleftarrow$ InferDurationTime($r_x$, $\hat{\phi}$, $G_R$)
6          if $r_x$ has been observed in an m-semantics $\lambda_x$ then
7              $\Lambda_o \longleftarrow \Lambda_o \setminus \lambda_x$; $\tau_x \longleftarrow \tau_x \cup \lambda_x.\tau$
8              $\Lambda_o \longleftarrow \Lambda_o \cup (r_x, \tau_x, \lambda_x.\delta)$
9          else
10             $\Lambda_o \longleftarrow \Lambda_o \cup (r_x, \tau_x, pass)$
11 return $\Lambda_o$
```

Appendix B FORMALIZATION DETAILS OF M-SEMANTICS INFERENCE

B.1 The Length of an Indoor Candidate Path

Lemma B.1. *Given an indoor path $\phi = r_i \rightarrow \ldots, \rightarrow r_j$, its path length $L(\phi) = \sum_{k=i}^{j} dist_{gr}(r_k, r_{k+1})$ is an* upper bound *of the minimum distance required to ensure any object can reach r_j from r_i through the path ϕ.*

Proof. Suppose that a region r_1 connects with region r_2 and the path corresponding to the GRD from r_1 to r_2 begins at a position $l_1^{(s)} \in r_1$ and ends at a position $l_2^{(e)} \in r_2$. Also, r_2 connects with region r_3 and the path corresponding to the GRD from r_2 to r_3 begins at a position $l_2^{(s)} \in r_2$ and ends at a position $l_3^{(e)} \in r_3$. Provided that an object o at the farthest position $l'_1 \in r_1$ from region r_3 can go through r_2 to reach r_3 at a position $l'_3 \in r_3$, and the corresponding traveling distance is $dist_I(l'_1, l'_2) + dist_I(l'_2, l'_3)$, where $l'_2 \in r_2$ is a position where o went out of r_1 and entered into r_2. According to the definition of GRD, we have $\forall l_1 \in r_1$, $dist_I(l_1, l_2^{(e)}) \leq dist_I(l_1^{(s)}, l_2^{(e)})$ and $\forall l_2 \in r_2$, $dist_I(l_2, l_3^{(e)}) \leq dist_I(l_2^{(s)}, l_3^{(e)})$. Consequently, we have $dist_I(l'_1, l'_2) + dist_I(l'_2, l'_3) \leq dist_I(l'_1, l_2^{(e)}) + dist_I(l'_2, l_3^{(e)}) \leq dist_I(l_1^{(s)}, l_2^{(e)}) + dist_I(l_2^{(s)}, l_3^{(e)}) = dist_{gr}(r_1, r_2) + dist_{gr}(r_2, r_3)$.

In a more general case, given an indoor path $r_i \rightarrow \ldots, \rightarrow r_j$, we have the minimum required distance that ensures an object can reach r_j from r_i through the path is no greater than $\sum_{k=i}^{j} dist_{gr}(r_k, r_{k+1})$. The lemma is proved. □

B.2 Posterior Probability of an Indoor Candidate Path

Given an observed region pattern $PT^{(o)}$ and a candidate path $\phi = r_s^{\shortparallel} \rightarrow r_a^{\triangleright} \rightarrow \ldots \rightarrow r_b^{\triangleright} \rightarrow r_q^{\triangleright} \rightarrow r_c^{\triangleright} \rightarrow \ldots \rightarrow r_d^{\triangleright} \rightarrow r_e^{\shortparallel}$, we have the posterior probability $\mathsf{P}(\phi|PT^{(o)})$ in the context of a first-order Markov stochastic process [30] as

$$\begin{aligned}
\mathsf{P}(\phi|PT^{(o)}) &= \mathsf{P}(r_s^{\shortparallel}, r_a^{\triangleright}, \ldots, r_b^{\triangleright}, r_q^{\triangleright}, r_c^{\triangleright}, \ldots, r_d^{\triangleright}, r_e^{\shortparallel} | r_s^{\shortparallel}, r_q^{\triangleright}, r_e^{\shortparallel}) \\
&= \mathsf{P}(r_s^{\shortparallel}, r_a^{\triangleright}, \ldots, r_b^{\triangleright}, r_q^{\triangleright} | r_s^{\shortparallel}, r_q^{\triangleright}) \cdot \mathsf{P}(r_q^{\triangleright}, r_c^{\triangleright}, \ldots, r_d^{\triangleright}, r_e^{\shortparallel} | r_q^{\triangleright}, r_e^{\shortparallel}) \\
&= \frac{\mathsf{P}(r_q^{\triangleright}|r_b^{\triangleright}) \prod_{x=a}^{b-1} \mathsf{P}(r_{x+1}^{\triangleright}|r_x^{\triangleright}) \mathsf{P}(r_a^{\triangleright}|r_s^{\shortparallel}) \mathsf{P}(r_s^{\shortparallel})}{\mathsf{P}(r_s^{\shortparallel}) \mathsf{P}(r_q^{\triangleright})} \cdot \frac{\mathsf{P}(r_e^{\shortparallel}|r_d^{\triangleright}) \prod_{y=c}^{d-1} \mathsf{P}(r_{y+1}^{\triangleright}|r_y^{\triangleright}) \mathsf{P}(r_c^{\triangleright}|r_q^{\triangleright}) \mathsf{P}(r_q^{\triangleright})}{\mathsf{P}(r_q^{\triangleright}) \mathsf{P}(r_e^{\shortparallel})} \\
&= \frac{\mathsf{P}(r_q^{\triangleright}|r_b^{\triangleright}) \prod_{x=a}^{b-1} \mathsf{P}(r_{x+1}^{\triangleright}|r_x^{\triangleright}) \mathsf{P}(r_a^{\triangleright}|r_s^{\shortparallel})}{\mathsf{P}(r_q^{\triangleright})} \cdot \frac{\mathsf{P}(r_e^{\shortparallel}|r_d^{\triangleright}) \prod_{y=c}^{d-1} \mathsf{P}(r_{y+1}^{\triangleright}|r_y^{\triangleright}) \mathsf{P}(r_c^{\triangleright}|r_q^{\triangleright})}{\mathsf{P}(r_e^{\shortparallel})} \\
&\propto \mathsf{P}(r_q^{\triangleright}|r_b^{\triangleright}) \prod_{x=a}^{b-1} \mathsf{P}(r_{x+1}^{\triangleright}|r_x^{\triangleright}) \mathsf{P}(r_a^{\triangleright}|r_s^{\shortparallel}) \cdot \mathsf{P}(r_e^{\shortparallel}|r_d^{\triangleright}) \prod_{y=c}^{d-1} \mathsf{P}(r_{y+1}^{\triangleright}|r_y^{\triangleright}) \mathsf{P}(r_c^{\triangleright}|r_q^{\triangleright})
\end{aligned}$$

Appendix C ADDITIONAL EXPERIMENTAL RESULTS

C.1 Answering Queries using M-Semantics on Real Data

In addition to the query time interval QT we evaluated in Section 6.1.5, we also investigate the effects of other parameters related to the two top-k queries, namely k and the query set size $|Q|$.

The parameter settings are shown in Table 5, where default values are in bold. We test each parameter with others fixed to defaults. The experimental evaluations focus on the search effectiveness in terms of the metric precision (see Section 6.1.5).

Table 5. Parameter Settings on Real Data

Parameters	Settings
Query Type	TkPRQ, TkFRPQ
QT (minutes)	60, **120**, 180, 240
k	20, 40, **60**, 80
$\|Q\|$ (% of semantic regions)	30%, **50%**, 70%

Effect of k. We fix $|Q| = 202 \times 50\% = 101$ and QT = 120 min, and vary k from 20 to 80. The results for TkPRQ and TkFRPQ are reported in Figure 16(a) and (b), respectively. As shown in Figure 16(a), the precisions of all search methods increase moderately with an increasing k. Since $|Q|$ is fixed in our test, a larger k tends to include more query regions in the top-k search results and therefore the precision improves in all the methods. Clearly, IMS-CA$\underline{\text{C}}$ performs the best in each k values; its precision stays higher than 0.84 with k up to 60. The results also verify the effectiveness of our cleaning method as the search methods with the cleaning (i.e., RAW-C, IMS-CA, and IMS-CA$\underline{\text{C}}$) clearly outperform other alternatives.

On the other hand, when we increase k, the precisions of all search methods stay very stable in processing the TkFRPQ query. Different from the TkPRQ, TkFRPQ needs to find k frequent region pairs from $|2^Q|$ candidate region pairs, whose number is significantly larger than that of the candidate regions in TkPRQ. In such a case, increasing k from 20 to 60 does not affect the precisions in all methods as k is relatively small compared to $|2^Q|$. Nevertheless, the m-semantics constructed by IMS-CA$\underline{\text{C}}$ are still the best in answering the TkFRPQ in different Q settings.

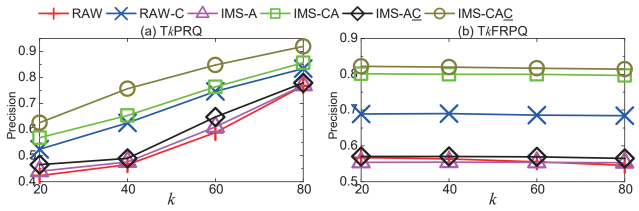


Fig. 16. Query Answering Effectiveness vs. k on Real Data

Effect of $|Q|$. We also very $|Q|$ from 30% to 70% with other parameters fixed by default. The results are reported in Figure 17. Referring to Figure 17(a), increasing $|Q|$ deteriorates the TkPRQ's precision in all the methods as more query regions need to be computed and ranked. However, the precision of our overall framework IMS-CA$\underline{C}$ decreases slightly compared to other alternatives. This shows that its constructed m-semantics are very effective to answer the top-k frequent region query. When increasing $|Q|$, the precisions of those methods without data cleaning decreases more rapidly than the alternatives that process on the cleaned data.

On the other hand, TkFRPQ's precision in each method is also insensitive to an increasing $|Q|$. In the shopping mall where our data was collected, the most frequent region pairs are always among the most popular stores visited by the shoppers. Hence, involving more semantic regions will not affect the returned results when k is fixed in the TkFRPQ query.

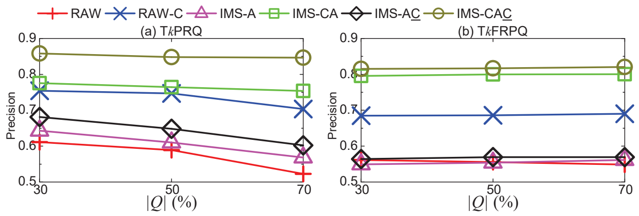


Fig. 17. Query Answering Effectiveness vs. $|Q|$ on Real Data

In general, the precision results on varying k and $|Q|$ demonstrate that the m-semantics constructed by our framework IMS-CA$\underline{C}$ are very effective in answering the two indoor top-k queries.

C.2 Effectiveness of Raw Data Cleaning on Synthetic Data

According to the ground truth trajectory recorded for each object, here we study the effectiveness of our proposed raw data cleaning method on the synthetic data.

Alternative Methods. To verify the effectiveness of using the indoor mobility constraints, we consider two alternatives, namely the ED method that uses the Euclidean distance to represent the speed constraint as well as interpolate the new location estimates, and the MIWD method that uses the MIWD instead. We apply different speed threshold v_m to ED and MIWD, and compare them with the original raw p-sequence without the cleaning (denoted as Original).

Performance Metrics. We measure the effectiveness of ED and MIWD in two aspects. On the one hand, given a p-sequence Θ and its ground truth Θ_g, we define Θ's *floor accuracy* as the fraction of its records having a correct floor value with respect to the ground truth. On the other hand, excluding those records that have false floor values, we define Θ's *average location error* (ALE) as

$$ALE = \sum\nolimits_{\theta \in \Theta, \theta_g \in \Theta_g, \theta.t = \theta_g.t, \theta.l.f = \theta_g.l.f} \frac{dist_I(\theta.l, \theta_g.l)}{|\Theta|}$$

For each synthetic IPT instance (see Table 4 in the paper), we measure the average value of floor accuracies and ALEs of all p-sequences. We vary and test different settings of the maximum positioning period T and the positioning error factor μ.

Effect of T. First, we fix $\mu = 3m$ and vary T from 5s to 15s. Referring to Figure 18(a), in each setting of T, the floor accuracy staying around 0.7 in the original p-sequence is improved significantly by the MIWD based methods. When increasing T, both methods' floor accuracy decreases, but MIWD's decreases slower than ED's. The decline is due to that the time difference between consecutive records becomes larger and the speed checking tends to be less reliable. Nevertheless, when $T = 15s$, MIWD still achieves a floor accuracy of around 0.9. On the other hand, MIWD beats ED in all tests when they use the same v_m. Interestingly, when we set v_m to $1.9m/s$ and vary T from 10s to 15s, both methods' floor accuracies increase, showing that a tighter speed constraint is better to capture object movements when data is sparser.

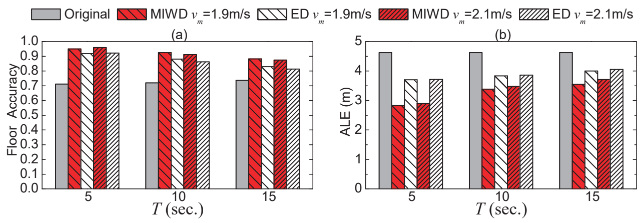


Fig. 18. Cleaning Effectiveness vs. T on Synthetic Data

Referring to Figure 18(b), the ALE of the original p-sequence is also reduced clearly by our cleaning methods. The reduction decreases but at a slow pace when a larger T is involved. Still, MIWD beats ED in all tests when we vary T.

The results reported on both measures verify that our mobility constraint based on MIWD is very effective in cleaning the raw indoor positioning data even when the data is temporally sparse.

Effect of μ. We vary and test the positioning error factor μ from 3m to 5m with T fixed to 5s. Referring to Figure 19(a), the floor accuracy is always improved significantly in different settings of μ. Also, MIWD's improvement only decreases slightly when μ increases. As reported in Figure 19(b), the ALE is also reduced clearly by the cleaning methods with different μs, and the reduction is more significant when μ increases. In both two measures, MIWD performs better than ED, showing it has a better capability to identify the errors and interpolate new location estimates.

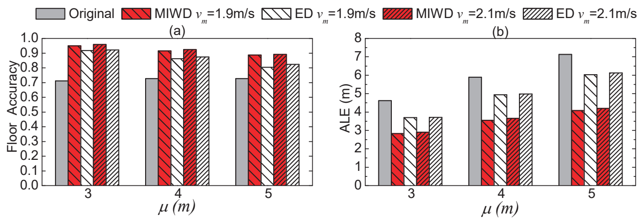


Fig. 19. Cleaning Effectiveness vs. μ on Synthetic Data

To sum up, the results in different settings of T and μ verify that our indoor mobility constraint based cleaning method using MIWD is still effective when the temporal sparsity and positioning errors are involved in the raw positioning data.

Received Aug 2019; revised Nov 2019; accepted Jan 2020