Towards Translating Raw Indoor Positioning Data into Mobility Semantics

Huan Li† Hua Lu‡ Gang Chen† Ke Chen† Qinkuang Chen† Lidan Shou†
† Department of Computer Science, Zhejiang University, China
‡ Department of Computer Science, Aalborg University, Denmark
{lihuancs, cg, chenk, chenqk, should}@zju.edu.cn, luhua@cs.aau.dk

ABSTRACT

Indoor mobility analyses are increasingly interesting due to the rapid growth of raw indoor positioning data. 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 describes a moving object's mobility event (What) at someplace (Where) at some time (When). The problem is non-trivial mainly because of the inherent errors in the uncertain, discrete raw data. To solve the problem, we propose a three-layer framework in which each layer contains a set of novel techniques. In the cleaning layer, we design a cleaning method that eliminates indoor 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. It includes a density based method that splits positioning sequences according to underlying mobility events and a semantic matching method that makes proper annotations for split snippets. In the complementing layer, we devise an inference method that makes uses of indoor topology and mobility semantics already obtained to recover the missing mobility semantics that are not observed in the raw data. 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.

PVLDB Reference Format:
Huan Li, Hua Lu, Gang Chen, Ke Chen, Qinkuang Chen and Lidan Shou. Towards Translating Raw Indoor Positioning Data into Mobility Semantics. *PVLDB*, 11 (3): xxxx-yyyy, 2017.
DOI: https://doi.org/TBD

1. INTRODUCTION

According to multiple studies [21, 25], people spend about 90% of their daily lives indoors. Human movements indoors are increasingly captured in raw positioning data due to the recent advance in indoor positioning [18] and high penetration of smartphones [2]. Analyzing indoor positioning data can reveal interesting findings otherwise hard to obtain, e.g., the popular indoor locations [30, 28] or hot routes [12, 36], frequent or exceptional indoor patterns [26, 20, 39], and insights for in-store marketing [13, 19].

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

In this study, we use a type of data generated by indoor positioning systems based on wireless technology (e.g., Wi-Fi) [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 (o, x, y, f, t) means the object o is estimated to be at 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 semantic trajectories [35, 46], we use indoor mobility semantics exemplified as follows:

o_1 : (*stay*, *Nike*, *1:02pm-1:18pm*) → (*pass-by*, *Adidas*, *1:19pm-1:20pm*) → (*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-by* 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.

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 very 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, 19]. Figure 1 depicts some Wi-Fi based positioning data obtained in a shopping mall in Hangzhou, China. Referring to Figure 1(a), for a device sampled every 10 seconds,

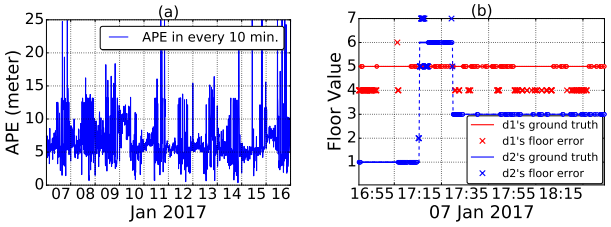

Figure 1: Real-world Indoor Positioning Data Errors

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

Proceedings of the VLDB Endowment, Vol. 11, No. 3

DOI: https://doi.org/TBD

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 produced. Second, indoor space accommodates multiple entities like rooms, doors, and barriers (e.g., walls), which altogether makes the object movements complex and hard to annotate. The situation becomes even worse when data errors are involved. Third, indoor positioning data is always discrete as mobile devices tend to turn off wireless signals for energy-saving needs [19]. 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) in a plane 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 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.*

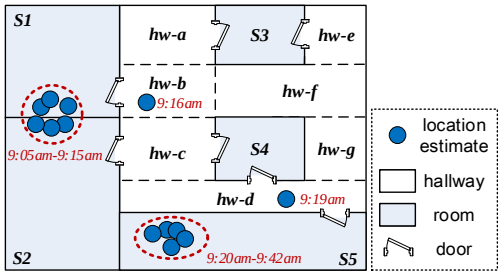

Figure 2: Example of Indoor Floorplan

To address the aforementioned challenges, we propose a three-layer framework for constructing m-semantics from the raw indoor positioning data. 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 the 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, in order 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 the top of the constructed knowledge. Consequently, each object's m-semantics sequence is complemented.

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

- 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 on the cleaned data. It includes a density based method that splits the positioning sequence according to the underlying mobility event and a semantic matching method that makes proper 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; Section 8 concludes the paper and discusses future work.

2. PRELIMINARIES

Table 1 lists the notations used throughout this paper.

Symbol	Meaning
o	an indoor moving object
$\theta = (o, l, t)$	object o's positioning record
Θ_o	object o's positioning sequence
$r \in \mathcal{R}$	an indoor (semantic) region
$\tau = [t_s, t_e]$	a time period
$\delta \in \{stay, pass\text{-}by\}$	a generic mobility event
$\lambda = (\pi, \tau, \delta)$	an indoor mobility semantics
$\Lambda_o = \langle \lambda_i, \ldots, \lambda_j \rangle$	object o's ms-sequence
$PT = \langle r_i, \ldots, r_j \rangle$	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
$dist_{gr}(r_s, r_e)$	Guaranteed Reaching Distance

Table 1: Notations

2.1 Raw Indoor Positioning Data

In our setting, an indoor positioning system aperiodically reports a record $\theta = (o, l, t)$ for an object o, where l is a location estimate and t is a timestamp, meaning that object o's location is estimated to be l at time t. In most indoor positioning systems [11, 17, 27], $\theta.l$ 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 indoor positioning records are stored in an *indoor positioning table* (IPT), as exemplified in Table 2.

o	l(x, y, f)	t
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

Table 2: Example of IPT

We define *indoor positioning sequence* (p-sequence) as follows.

DEFINITION 1 (INDOOR POSITIONING SEQUENCE). *Given an IPT and a time period $\mathcal{T} = [t_s, t_e]$, an object o's indoor positioning sequence over $\mathcal{T}$ is a complete time-ordered sequence $\Theta_{o,\mathcal{T}}$ of indoor positioning records of o. At most one positioning record (o_i, l_i, t_i) can be found in $\Theta_{o,\mathcal{T}}$ for a unique timestamp t_i.*

Referring to Table 2, object o_1's p-sequence over $\mathcal{T} = [t_1, t_4]$ is $\langle (o_1, (2.5, 10.7, 1), t_1), (o_1, (2.3, 11.2, 2), t_4) \rangle$. When the context is clear, we use Θ_o to denote object o's p-sequence.

2.2 Problem Statement

We formally give the definition of *mobility semantics*.

DEFINITION 2 (MOBILITY SEMANTICS). *Corresponding to a segment of p-sequence $\Theta_o^* \subseteq \Theta_o$ of an indoor object o, a* mobility semantics *(m-semantics for short) is a triplet $\lambda(\pi, \tau, \delta)$, where the spatial annotation π is an indoor region, the temporal annotation τ is a time period $[t_i, t_j]$ and the event annotation δ is a generic mobility event given by an identification function $\mathcal{E}(\Theta_o^*)$.*

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 naturally divided into a number of

indoor partitions like rooms and hallways by the walls and doors. For simplicity, we assume that each indoor region is composed of one or more such indoor partitions [2]. When the context is clear, we use regions to refer to indoor regions.

The *mobility events* refer to some interesting movement patterns. Exisiting patterns like stop/move [46] have been used to describe the movements in geographic space, e.g., a car moves on a road or a person stops at a park. Compared to the geographic space, an indoor space is composed of much smaller and denser indoor regions that fullfil different purposes (e.g., a carteen vs. a meeting room). Using stop/move does not well reflect the underlying purpose of the movements in or between the semantic-rich indoor regions. To give more informative semantics, we introduce two generic indoor mobility events *stay* and *pass-by*. In particular, a stay indicates that an indoor object has been *staying in* a region for a time period for a particular purpose that is fulfilled in that region. E.g., a stay can be that a user spent half hour in a shoe shop, selecting and buying a pair of new shoes. In contrast, a pass-by tells that an object has *passed through* a region but there are no particular purposes associated with that pass. E.g., a user might pass by a number of other shops before she reaches the shop where she bought shoes. The distinction between stay and pass-by is very useful in pertinent indoor scenarios. For example, security managers may only have interest in the people staying in a certain indoor region, while the mall managers may want to know both numbers of staying and passing-by customers when analyzing advertising effects 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-by. The $\mathcal{E}$-function is a learning-based model and supervised with the spatiotemporal features extracted from the user-recognized mobility events. An m-semantics annotated with a stay (pass-by) event is a stay (pass-by) m-semantics and denoted by $\lambda^{\shortparallel}$ ($\lambda^{\triangleright}$), and its associated region is a stay (pass-by) region denoted as $r^{\shortparallel}$ ($r^{\triangleright}$).

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

DEFINITION 3 (M-SEMANTICS SEQUENCE). *Given an indoor object o, its* m-semantics sequence *over a time period $\mathcal{T}$ is a time-ordered sequence $\Lambda_{o,\mathcal{T}}$ of o's m-semantics. For any m-semantics $\lambda_i, \lambda_j \in \Lambda_{o,\mathcal{T}}$, $\lambda_i.\tau \subseteq \mathcal{T}, \lambda_j.\tau \subseteq \mathcal{T}, \lambda_j.\tau \cap \lambda_j.\tau = \varnothing$.*

We formulate our research problem as follows.

PROBLEM 1 (INDOOR M-SEMANTICS CONSTRUCTION). *Given an IPT, a time period $\mathcal{T}$, and a set $\mathcal{R}$ of indoor regions, for each object o_i in IPT, the* indoor m-semantics construction *translates o_i's p-sequence $\Theta_{o_i,\mathcal{T}} = \langle \theta_{i1}, \ldots, \theta_{in} \rangle$ into an ms-sequence $\Lambda_{o_i,\mathcal{T}} = \langle \lambda_{i1}, \ldots, \lambda_{im} \rangle$.*

Constructing ms-sequences provides an intuitive, concise way to understand an indoor moving object's general 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.

2.3 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 from each individual object. It conducts the data cleaning by considering the indoor mobility constraints captured in a distance-aware model.

[2]Assuming regions do not overlap, our techniques allow an indoor region to involve (parts of) an indoor partition.

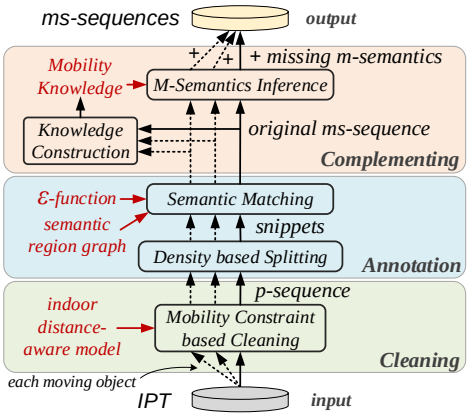

Figure 3: The Construction Framework

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 an $\mathcal{E}$-function and a *semantic region graph*.

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 the mobility knowledge.

3. RAW POSITIONING DATA CLEANING

As shown in Figure 1, raw indoor positioning data contains inherent errors due to the limitations of 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 [27].
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 in 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 speaking, indoor object movements should comply with relevant *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 $\langle (o, l_1, t_1), (o, l_2, t_2), (o, l_3, t_3) \rangle$, 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 estimate l_2 is outside the shaded part at time t_2, l_2 is an error.

When computing the object moving speed between two positioning locations, we use the *indoor distance-aware model* [29] that supports to compute the *minimum indoor walking distance*

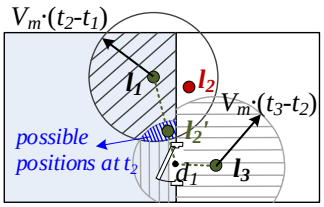

Figure 4: Example of Cleaning Raw Indoor Positioning Data

(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 a forward direction, a record θ_i that violates the aforementioned speed constraint should be invalid if its preceding record θ_{i-1} has already been determined as valid.

When an invalid record θ_i is identified, the error in its location estimate may occur at the floor part (i.e., false floor value) and/or the 2D point part (i.e., location outlier or random error). We repair it in two steps. The first step repairs a potential mistake in θ_i's 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 underlying data error is still in the 2D point part. The second step repairs the wrong location estimate by interpolating a new one. To this end, 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 how the corresponding object has moved 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 indoor 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 intersection 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 part. We omit the details due to the page limit.

The overall cleaning is formalized in Algorithm 1. For simplicity, we assume that the first element in the p-sequence is valid. Nevertheless, the algorithm can start with any valid positioning record that results from a global checking on the p-sequence.

Algorithm 1: Mobility Constraint based Cleaning

Input: p-sequence Θ_o, maximum moving speed V_m.
Output: *cleaned* p-sequence Θ_o'.

```
1  time-ordered sequence A_θ ⟵ ⟨⟩
2  current valid positioning record θ̂ ⟵ head(Θ_o)
3  for each positioning record θ ∈ Θ_o \ head(Θ_o) do
4      valid ⟵ True
5      if ValidSpeed(θ̂, θ) is False then
6          (x, y, f) ⟵ θ.l; (x̂, ŷ, f̂) ⟵ θ̂.l; θ.l ⟵ (x, y, f̂)
7          if ValidSpeed(θ̂, θ) is False then
8              add θ to A_θ; valid ⟵ False
9      if valid then
10         A'_θ ⟵ Interpolation(θ̂, A_θ, θ)
11         add all positioning records in A'_θ to Θ'_o
12         A_θ ⟵ ⟨⟩; θ̂ ⟵ θ
13 add θ̂ to Θ'_o
14 return Θ'_o
15 Function ValidSpeed(θ̂, θ)
16     v ⟵ dist_I(θ̂.l, θ.l) / (θ.t - θ̂.t)  // compute the MIWD
17     if v ≤ V_m then return True else return False
18 Function Interpolation(θ_p, A_θ, θ_s)
19     A'_θ ⟵ ⟨⟩; add θ_p to A'_θ
20     compute a shortest indoor path pa from θ_p.l to θ_s.l
21     for each positioning record θ' ∈ A_θ do
22         find a location l on pa having
               dist_I(θ_p.l, l) = (θ'.t - θ_p.t)/(θ_s.t - θ_p.t) · dist_I(θ_p.l, θ_s.l)
23         θ'.l ⟵ l; add θ' to A'_θ
24     return A'_θ
```

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.

Algorithm 2: Split-and-Match Annotation

Input: p-sequence Θ_o, event identification function $\mathcal{E}$, semantic region graph $G_{\mathcal{R}}$.
Output: time-ordered sequence of m-semantics Λ_o.

```
1  time-ordered sequence Λ_o ⟵ ⟨⟩
2  A_snpt ⟵ DensityBasedSplitting(Θ_o)
3  for each snippet Θ*_o in A_snpt do
4      Λ*_o ⟵ SemanticMatching(Θ*_o, E, G_R)
5      Λ_o ⟵ Λ_o ∪ Λ*_o
6  return Λ_o
```

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-by introduced in Section 2.2). Next, in each split snippet, we match its m-semantics

4.1 Density Based Splitting

As an object's underlying mobility events are a sequence of mixing *stay* or *pass-by*, a natural splitting way is to divide the p-sequence into periods of time when the object is considered as relatively stationary (i.e., stay) and periods when the object is moving (i.e., pass-by). The SMoT method [4] for GPS data partitions the GPS sequence based on the instant speed between two consecutive records. Given a speed threshold ψ, the period in which the instant speed is lower than ψ is considered as a stay event. In contrast, the lasting period with instant speeds greater than ψ is regarded as a pass-by event. However, SMoT is insufficient to process indoor p-sequences. On the one hand, indoor movements are relatively slow, making it hard to decide a suitable ψ to distinguish stay and pass-by events. On the other hand, indoor positioning data usually features

a low sampling rate. As a result, the instant speed computed for two consecutive records cannot reflect the underlying movement [3].

Despite the speed information, the positioning records observed from a stay event always have both 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 data instances according to 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 for temporal attributes; 3) pt_m is a number 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.

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 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 [4].

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.*

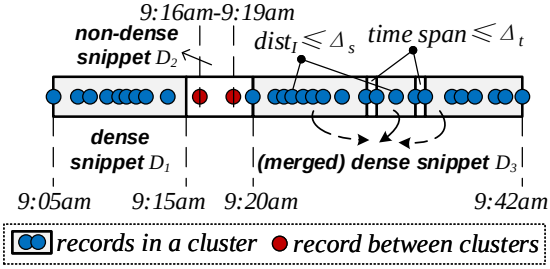

Figure 5: Example of Density based Splitting on a P-sequence

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-by. 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 one single place. To verify if a snippet corresponds to 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 an event identification technique in Section 4.2.1.

[3]Our cleaning method is not affected by this case since it only identifies a part of distinct data errors that violate the speed constraints.

[4]The appendices are available at https://goo.gl/zzriSD.

4.2 Semantic Matching

4.2.1 Event Identification Function

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

- *Dense Level.* As the positioning records of stay events usually fall in the formed clusters, it is useful to indicate whether a snippet is dense or not.
- *Variance of Location Estimates.* We consider the variance of all associated location estimates as it is usually very small in stay events but relatively large in pass-by events.
- *Sampling Conditions.* The record number and sampling rate in the snippet are calculated as the indoor positioning data is usually sparser when the object (wireless device) is moving.
- *Covering Range.* The geometric shape (i.e., convex hull or its simplified minimum boundary rectangle) that covers all location estimates of the snippet, along with its area and centroid.
- *Overlapping Regions.* The IDs of the indoor regions that cover or intersect with the covering range, along with each such region's relevant record number (only the top-n regions are used).
- *Walking Distance.* The sum and average of the MIWDs between every two consecutive location estimates.
- *Walking Speed.* The maximum, minimum and average of the instant speeds between every two consecutive records.
- *Number of Turns.* Studies reveal that people only make a very small number of turns when they are walking indoors [36]. Thus, the ratio between the number of turns and walking distance is considered in order to identify the pass-by events.

Next, a logistic regression model [3] is employed to classify the stop and pass-by events. To train the model, a feature set is extracted from the snippets labeled with *stop* or *pass-by*. A co-training method [9] is introduced to iteratively construct additional labeled training data when only small amounts of labeled data are available. We omit the low-level details due to the page limit.

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

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-by 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-by 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-by m-semantics. In the following, we introduce a *semantic region graph* $G_{\mathcal{R}}$ that facilitates accessing a set $\mathcal{R}$ of indoor regions specified by user semantics (see Section 2.2). Specifically, $G_{\mathcal{R}}$ is a labeled, directed graph represented in a 5-tuple $(V, E, G_{dist}, R2P, P2R)$:

1) Each vertex $v \in V$ is an indoor region $r \in \mathcal{R}$.
2) E is the edge set $\{\langle v_i, v_j, \mathbb{R}\rangle \mid v_i, v_j \in V\}$. Each directed edge gives the *guaranteed reaching distance* (GRD) from an indoor region v_i to another *directly connected* [5] indoor region v_j.
3) G_{dist} is the associated distance-aware model [29] that involves with indoor entities like doors and partitions (see Section 3).
4) $R2P : \mathcal{R} \rightarrow 2^{\mathcal{P}}$ maps an indoor region (vertex) to the set of indoor partitions in G_{dist} it contains.
5) $P2R : \mathcal{P} \rightarrow \mathcal{R}$ maps an indoor partition in G_{dist} to the indoor region that covers it.

Given two indoor regions r_i, r_j and r_j's enterable door set [6] $P2D_{\sqsupset}(R2P(r_j))$, the **guaranteed reaching distance** (GRD) from r_i to r_j is defined as

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

Generally, the GRD from r_i to r_j is the walking distance an indoor object need to reach r_j from a 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-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.

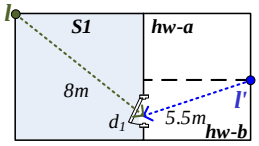

Figure 6: GRD Example

Figure 7 gives the graph $G_{\mathcal{R}}$ corresponding to Figure 2. By using the $R2P$ and $P2R$ mappings, a region *hw-f* is mapped to two partitions it contains, and a partition p_{13} is mapped to its covering region *hw-e*. Also, it can be seen that an object currently in *S2* can reach *hw-c* within a distance $dist_{gr}(S2, hw\text{-}c) = 9.5m$.

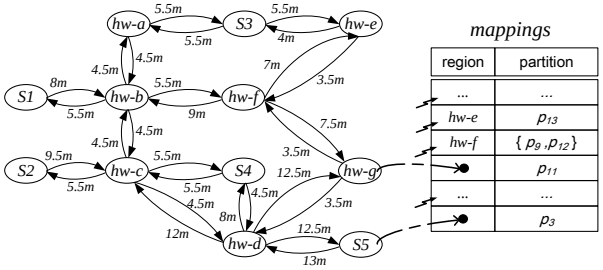

Figure 7: Example of Semantic Region Graph $G_{\mathcal{R}}$

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 estimate's intersected partition is found via the R-tree, the relevant covering region can be obtained via the mappings defined in $G_{\mathcal{R}}$. Note that these mappings allow users to specify the indoor regions without getting to the underlying spatial searching conducted on the indoor partitions level. Consequently, we can use $G_{\mathcal{R}}$ 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-by and stay m-semantics. For a pass-by semantics and its corresponding positioning record θ, we simply consider the indoor region that contains $\theta.l$ as the spatial annotation. The consecutive pass-by 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. The traditional methods use the centroid of location estimates or a voting mechanism to decide the region where the object has been staying. Such methods assume each location estimate is independently observed and has the same importance to determine the underlying stay position of the object.

Actually, an object staying in a place usually incurs very small displacements between its consecutive location estimates. If an estimate is fairly far away from its neighboring estimates, it should be affected by the positioning random errors and thus is less reliable. Motivated as such, we propose a concept of *location estimate confidence* that gives different weights to the location estimates in determining the object's stay position.

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 $\theta_i.l$ ($s \leq i \leq e$) is defined as*

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

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 $\omega^{(i)} = \frac{conf^{(i)}}{\sum_{j=s}^{e} conf^{(j)}}$ and infer the underlying stay position as $\hat{l} = \sum_{i=s}^{e} \omega^{(i)} \cdot \theta_i.l$. As a result, the spatial annotation for a stay m-semantics is made as a region $r \in \mathcal{R}$ 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-by and stay, respectively. Refer 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-by m-semantics, (hw-b, 9:16am-9:16am, pass-by) and (hw-d, 9:19am-9:19am, pass-by).*

5. COMPLEMENTING MS-SEQUENCES

In an ms-sequence obtained from the annotation layer, pass-by m-semantics may be temporally very discrete, implying there may be information between the associated positioning records. To make such an ms-sequence complete and coherent, we proceed to present an inference method to recover the missing m-semantics.

Multiple studies [26, 36] have discovered that people often make very similar movements between two indoor destinations within

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

[6] In the distance-aware model [29], the mapping $P2D_{\sqsupset}$ gives the enterable doors for a given indoor partition.

a relatively small range, regardless of people's walking purposes (e.g., to find something or just to look around somewhere). This finding inspires us to make use of such similar movements to infer the missing m-semantics between two relevant regions. Specifically, each stay region $r^{\shortparallel}$ in an annotated stay m-semantics can be considered as an indoor destination, and those annotated pass-by m-semantics between two destinations can be grouped together to capture the similar movements in-between. By further considering the indoor mobility constraints, we are therefore able to infer the unobserved movements given a sequence of m-semantics we have 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 any two stay regions (i.e., destinations). Using the constructed knowledge, the second phase (lines 2–5) 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 $G_\mathcal{R}$.
Output: set of *complemented* ms-sequences $\mathcal{S}'_\Lambda$.

1 set $\mathcal{S}'_\Lambda \longleftarrow \varnothing$
2 hash table $\mathcal{MK} \longleftarrow$ *ConstructMobilityKnowledge*($G_\mathcal{R}, \mathcal{S}_\Lambda$)
3 **for** each original ms-sequence Λ_o in $\mathcal{S}_\Lambda$ **do**
4 $\Lambda_o \longleftarrow$ *MSemanticsInference*($\Lambda_o, \mathcal{MK}, G_\mathcal{R}$)
5 add Λ_o to $\mathcal{S}'_\Lambda$
6 **return** $\mathcal{S}'_\Lambda$

5.1 Mobility Knowledge Construction

Given two stay regions $r_s^{\shortparallel}, r_e^{\shortparallel}$, the *mobility knowledge* about the similar movements from $r_s^{\shortparallel}$ to $r_e^{\shortparallel}$ 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 $G_\mathcal{R}$. The second part is the transition probabilities between the directly connected regions in the candidate paths. Next, we elaborate on how to construct the candidate path set and the transition probabilities, respectively.

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

DEFINITION 5 (INDOOR CANDIDATE PATH). *Given a start region $r_s^{\shortparallel}$ and an end region $r_e^{\shortparallel}$, a candidate path from $r_s^{\shortparallel}$ to $r_e^{\shortparallel}$ is a region sequence $\phi = r_s^{\shortparallel} \rightarrow r_i^{\triangleright} \rightarrow \ldots \rightarrow r_j^{\triangleright} \rightarrow r_e^{\shortparallel}$, each pass-by region $r_k^{\triangleright}$ $(i \leq k \leq j)$ contained in ϕ is* unique*, and each two consecutive regions in ϕ are* directly connected*. Moreover, ϕ's* path length $L(\phi)$ *is given by*

$$dist_{gr}(r_s^{\shortparallel}, r_i^{\triangleright}) + \sum_{k=i}^{j-1} dist_{gr}(r_k^{\triangleright}, r_{k+1}^{\triangleright}) + dist_{gr}(r_j^{\triangleright}, r_e^{\shortparallel}) \quad (3)$$

where $dist_{gr}$ is the GRD captured in $G_\mathcal{R}$.

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 1 in Appendix B.1.

To find a set P of candidate paths from $r_s^{\shortparallel}$ to $r_e^{\shortparallel}$, we perform an A*-Search [48] on $G_\mathcal{R}$. As the number of such candidate paths can be very large, we use a path length threshold γ to filter out those paths whose length is extremely long since the similar movements are within a relatively small range [26]. 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 indoor 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 $G_\mathcal{R}$.

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 29.5m, the shortest path length from S1 to S5. As a result, 4 indoor candidate paths can be searched from the $G_\mathcal{R}$ shown in Figure 7.*

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

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 $PT = \langle r_i^{\triangleright}, \ldots, r_j^{\triangleright} \rangle$ [7].

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 P' of candidate path set P, in which all paths hold PT.
2) For each path $\phi \in P'$, we compute its path length $L(\phi)$ according to Equation 3. As moving objects tend to choose a shorter path on move [36], each path ϕ's weight ω_ϕ among all possible paths is considered to be inversely proportional to $L(\phi)$ and computed as $\omega_\phi = L(\phi)^{-1} / \sum_{\phi \in P'} (L(\phi)^{-1})$.
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, $score(\langle r_k, r_{k+1} \rangle)$ += $PT.count * \omega_\phi$, where $PT.count$ is PT's count number in $\mathcal{H}_{PT}$.
4) After each PT has been processed, we compute the **transition probability** from region r_i to r_j as

$$P_t(r_i, r_j) = \frac{score(\langle r_i, r_j \rangle)}{\sum_{r \in Out(r_i)} score(\langle r_i, r \rangle)} \quad (4)$$

where $Out(r_i)$ is the set of all directly connected regions in $G_\mathcal{R}$ 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 [8]. 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 $G_\mathcal{R}$ for the sake of presentation. Refer to*

[7] We omit the start and end regions $r_s^{\shortparallel}$ and $r_e^{\shortparallel}$ if the context is clear.
[8] 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 in order to constrain the candidate path generation.

a region pattern $PT = \langle$hw-b, hw-c, hw-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-b$\rangle$ and $\langle$hw-b, hw-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.*

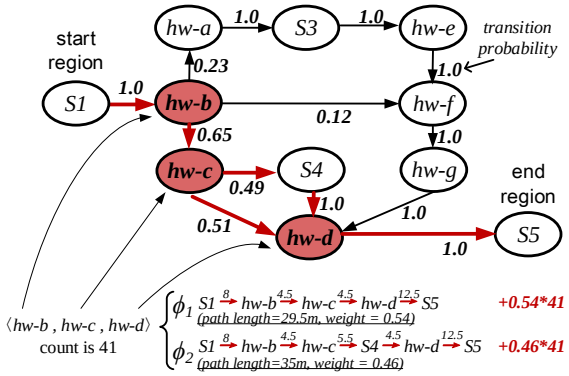

Figure 8: Example of Mobility Knowledge Construction

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 6.1.3.

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

Generally, for an observed ms-sequence $\Lambda_o^{(o)} = \langle \lambda_s^{\shortparallel}, \lambda_q^{\triangleright}, \lambda_e^{\shortparallel} \rangle$, our path inference needs to find a 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 P constructed for $\langle r_s^{\shortparallel}, r_e^{\shortparallel} \rangle$, each path $\phi \in P$ that supports $PT^{(o)}$ can be denoted as $r_s^{\shortparallel} \rightarrow r_a^{\triangleright} \rightarrow \ldots \rightarrow r_b^{\triangleright} \rightarrow r_q^{\triangleright} \rightarrow r_c^{\triangleright} \rightarrow \ldots \rightarrow r_d^{\triangleright} \rightarrow r_e^{\shortparallel}$, where $r_a^{\triangleright} \rightarrow \ldots \rightarrow r_b^{\triangleright}$ and $r_c^{\triangleright} \rightarrow \ldots \rightarrow r_d^{\triangleright}$ 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.

Following the literature [40, 32] 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* $P(\phi|PT^{(o)})$ [9] satisfies the expression below.

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

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

$$\arg\max_{\phi} P(\phi|PT^{(o)}) = \arg\max_{r_a^{\triangleright} \rightarrow \ldots \rightarrow r_b^{\triangleright} \subseteq \phi} P_t(r_s^{\shortparallel}, r_a^{\triangleright}) \prod_{x=a}^{b-1} P_t(r_x^{\triangleright}, r_{x+1}^{\triangleright}) P_t(r_b^{\triangleright}, r_q^{\triangleright}) \arg\max_{r_c^{\triangleright} \rightarrow \ldots \rightarrow r_d^{\triangleright} \subseteq \phi} P_t(r_q^{\triangleright}, r_c^{\triangleright}) \prod_{y=c}^{d-1} P_t(r_y^{\triangleright}, r_{y+1}^{\triangleright}) P_t(r_d^{\triangleright}, r_e^{\shortparallel}) \quad (6)$$

A *max-product* algorithm [16] is used to solve this problem. By taking the transition probabilities from the 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 those 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 P, we change it to the most adjacent region r'' from those contained by any path in P.

EXAMPLE 6. *Refer to the ms-sequence in Example 3. We obtain its observed region pattern $PT^{(o)} = \langle$S1, hw-b, hw-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 $\rightarrow$ hw-b, hw-b $\rightarrow$ hw-c $\rightarrow$ hw-d, and hw-d $\rightarrow$ S5, respectively. The most-likely path is then assembled as S1 $\rightarrow$ hw-b $\rightarrow$ hw-c $\rightarrow$ hw-d $\rightarrow$ S5.*

5.2.2 Time Period Inference

In the most-likely path inference, we find an optimal sub-path $\phi^* = r_p \rightarrow \ldots \rightarrow r_q$ for each two consecutive m-semantics λ_p and λ_q in an observed ms-sequence. Next, we annotate each region $r_x \in \phi^*$ with a temporal annotation, i.e., a time period, to fill 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., $\mathcal{T}_{\lambda_p,\lambda_q} = [\lambda_p.\tau.t_e, \lambda_q.\tau.t_s]$. However, it is hard to determine the time period that the object is in r_x as the object movement is unobserved during $\mathcal{T}_{\lambda_p,\lambda_q}$. 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 moving objects. To ease computation, we assume that the moving object moves along its path with a constant speed during $\mathcal{T}_{\lambda_p,\lambda_q}$. Consequently, we can use the GRD between two regions as the reference [10] to divide r_x's time period from $\mathcal{T}_{\lambda_p,\lambda_q}$.

Formally, for a region r_x in $\phi^* = r_p \rightarrow \ldots \rightarrow r_q$, its time period is inferred as $\tau_x = [t_s^{(x)}, t_e^{(x)}]$ where

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

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

When the time period τ_x is inferred for r_x, we differentiate two cases. If r_x has already appeared in an m-semantics in the observed

[9] Detailed derivation is given in Appendix B.2.

[10] As discussed in Section 4.2.2, a larger region usually costs more time to come out as indicated by its GRDs to its connected regions.

ms-sequence, the time period is added to the corresponding m-semantics. Otherwise, we should generate a missing m-semantics as $(r_x, \tau_x, \textit{pass-by})$, 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. As a result, the complementing is accomplished.

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-by). Likewise, we process other sub-paths and obtain a complete ms-sequence:*

$$(\textit{S1}, \textit{9:05am-9:16am}, \textit{stay}) \rightarrow (\textit{hw-b}, \textit{9:16am-9:18am}, \textit{pass-by})$$
$$\rightarrow (\textit{hw-c}, \textit{9:18am-9:19am}, \textit{pass-by}) \rightarrow (\textit{hw-d}, \textit{9:19am-9:20am}, \textit{pass-by})$$
$$\rightarrow (\textit{S5}, \textit{9:20am-9:42am}, \textit{stay})$$

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. We decomposed the whole indoor space [11] and 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 developed a visualization tool [12] to manually annotate m-semantics on the p-sequences as we were unable to know a device's exact whereabouts. Among the ms-sequences we annotated, 9,687 ms-sequences (including 125,544 m-semantics) formed the ground truth for evaluation, and the other 1,004 ms-sequences (including 17,322 m-semantics) were used to initialize the $\mathcal{E}$-function on Jan 1. Particularly, an individual stay m-semantics or a number of consecutive pass-by 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.

As the m-semantics is newly defined in this paper, we propose a new metric to measure its effectiveness with respect to ground truth. Formally, we say a constructed m-semantics λ is η-*acceptable* to its ground truth λ_g if $\lambda.\pi = \lambda_g.\pi$, $\lambda.\delta = \lambda_g.\delta$ and $\frac{|\lambda.\tau \cap \lambda_g.\tau|}{|\lambda_g.\tau|} \geq \eta$. A bigger η indicates that λ is more consistent with λ_g, and thus more accurate and qualified to satisfy the application need. E.g., applications in security usually require larger η-acceptable m-semantics than those in shopper analysis. Consequently, we define the η-*acceptable recall* as the fraction of ground truth m-semantics that can find an η-acceptable m-semantics among all ground truth m-semantics. When $\eta = 1$, our η-acceptable recall equals to the metric recall. We do not use the conventional recall because it is too rigid for m-semantics that hardly have exact, simultaneous spatial and temporal matches in the ground truth.

6.1.1 Comparison of Annotation Methods

Our annotation method, denoted as *Dense-$\mathcal{E}$+LEC*, was implemented as two key modules. One is a temporal-event annotator depending on the density based splitting method (Section 4.1) and $\mathcal{E}$-function (Section 4.2.1). The other is a 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.

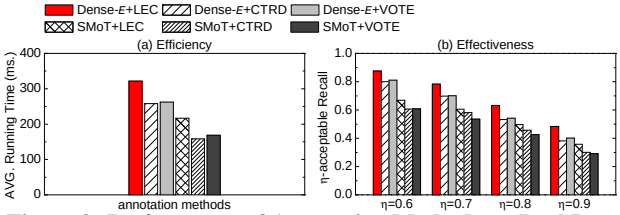

Figure 9: Performance of Annotation Methods on Real Data

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 η-acceptable recalls in different settings of η. The methods with Dense-$\mathcal{E}$ clearly outperform those with SMoT, which shows the spatiotemporal density information is more useful than the speed information in splitting p-sequences and identifying mobility events. Combining with either Dense-$\mathcal{E}$ or SMoT,

[11] The decomposition algorithm is given in [43].

[12] A relevant work is under review in the demo track. A demonstration video is available at https://longaspire.github.io/trips.

the spatial matching method LEC always outperforms VOTE and CTRD in all η values.

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-CAC* (C denotes cleaning and 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-AC* adds a complementing layer after the annotation layer. Note that the annotation layer is necessary for constructing m-mobility. We report the construction results on the number of m-semantics per ms-sequence and η-acceptable recall for these four methods in Table 3.

Method	Number of M-Semantics	η-acceptable Recall			
		η=0.6	η=0.7	η=0.8	η=0.9
IMS-A	11.94	0.3555	0.2926	0.2187	0.1642
IMS-CA	10.23	0.6615	0.5577	0.3825	0.2825
IMS-AC	14.51	0.4645	0.3858	0.2638	0.2155
IMS-CAC	14.12	**0.8756**	**0.7828**	**0.6318**	**0.4834**

Table 3: Effect of Cleaning and Complementing on Real Data

Clearly, IMS-A is the worst as it directly annotates m-semantics on the raw data. Only 35.6% of the ground truth can find a 0.6-acceptable m-semantics in the result of IMS-A. In contrast, IMS-CA with a cleaning layer significantly outperforms IMS-A; the recall in each setting is increased almost by double, and 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 is able to improve the input data for subsequent layers. This effect of cleaning is also observed when we compare the results of IMS-CA and IMS-AC. Although the latter produces more m-semantics, many of them are problematic as they result from uncleaned data. Therefore, the recalls of IMS-AC are considerably lower than their counterparts of IMS-CA.

IMS-CAC is always the best among all. When η is set to 0.6, its recall is greater than 0.87. Also, 48% of ground truth can be matched with a 0.9-acceptable m-semantics. These results show that IMS-CAC is able to produce reliable m-semantics highly consistent with the ground truth. With the help of our inference based complementing, IMS-CAC is able to 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 to 14.12 from 10.23. More importantly, because of the combined effects of cleaning and complementing, IMS-CAC's recall improves clearly compared to IMS-CA and the others. In summary, the m-semantics construction on real data remarkably benefit 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 layer 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 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 0.6-acceptable recalls for the IMS-CA and IMS-CAC methods in each day from Jan 2 to Jan 31. The methods without daily updating are marked with '*w/o DU*'. Referring to Figure 10, the recalls of the methods without daily updating fluctuate a lot and in general decrease as 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-CAC 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.

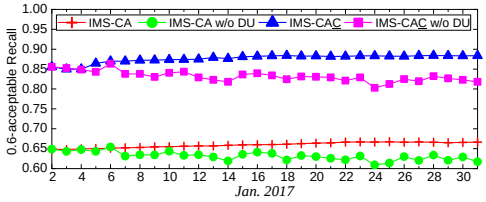

Figure 10: Effectiveness vs. Daily Updating on Real Data

6.1.4 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 time interval $\mathcal{T}$, 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* [13] within $\mathcal{T}$.
2) A *Top-k Frequent Region Pair Query* (TkFRPQ) finds k most frequent pairs of semantic regions from 2^Q that both have been visited by the same object within $\mathcal{T}$.

TkPRQ and TkFRPQ are very useful in studies like popular indoor location discovery [30] and frequent pattern mining [26, 20]. Other than the 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, an interpretation on *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 period of $\mathcal{T}_s$, the object is considered to have visited that region for once. We set $\mathcal{T}_s$ 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 $\mathcal{T}$ as 60, 120, 180, 240 min..

The efficiency results for TkPRQ and TkFRPQ are reported in Figure 11(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 $\mathcal{T}$ 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

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

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-CAC was only 220.1 MB. Moreover, IMS-CAC can return the results for both TkPRQ and TkFRPQ 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.

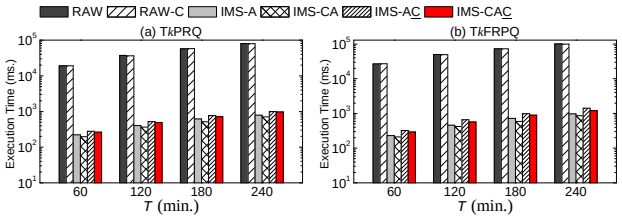

Figure 11: Query Answering Efficiency vs. $\mathcal{T}$ on Real Data

We also measured the effectiveness in each aforementioned setting. As shown in Figure 12(a) and (b), for both types of queries, the precision of all methods decreases with an increasing $\mathcal{T}$. When a longer $\mathcal{T}$ 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-CAC) decrease very slowly. Among them, IMS-CA and IMS-CAC always outperform RAW-C, showing that the brief information kept in m-semantics can capture the underlying object movements very well. Moreover, when $\mathcal{T}$ increases to 240 min., m-semantics constructed by IMS-CAC can still achieve a precision 82.8% for TkPRQ and 79.1% for TkFRPQ.

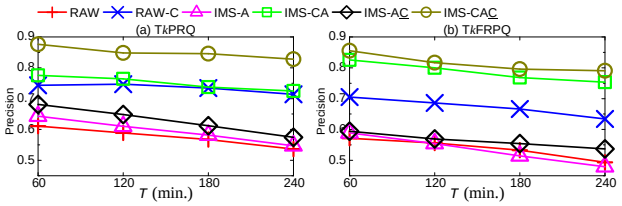

Figure 12: Query Answering Effectiveness vs. $\mathcal{T}$ on Real Data

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-CAC are significantly effective in answering the two 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* [27], 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 a period of 4 hours. Specifically, 10K objects were distributed to the floors, each having a lifespan varied from 10 seconds to 4 hours. Object maximum speed was set to $V_{max} = 1.7m/s$ and object movements followed the waypoint model [23]. In particular, each semantic region is considered as a destination, an object moves towards its destination along a preplanned indoor path, it stays at the destination for a random time 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-by) 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.

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

Table 4: Synthetic IPT Instances

The memory consumptions for $G_{\mathcal{R}}$, 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. We set a medium $\eta = 0.7$ to measure the consistency between constructed m-semantics and the ground truth described above. First, we fixed $\mu = 3m$ and vary T. The results are reported in Figure 13(a). When varying T from 5s to 15s, i.e., the observed data becomes sparser (see Table 4), and all methods' recalls decrease but IMS-CAC's decreases the slowest. Also, the performance gap between IMS-CAC 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-CAC can still have a 0.7-acceptable recall of 83%. Still, IMS-A's recall is the worst and decreases rapidly when T increases. IMS-CA and IMS-CAC equipped with a cleaning layer clearly outperform the other two in all tests.

We also fixed T to $5s$ and tested with different μs. Refer to the recall measures reported in Figure 13(b). When μ increases, both IMS-CAC and IMS-CA stay stable while the others without cleaning decrease rapidly. This demonstrate that our raw data cleaning is very useful to reduce the negative impact of the positioning errors. Besides, IMS-CAC outperforms IMS-CA in all tests due to the benefits of the complementing.

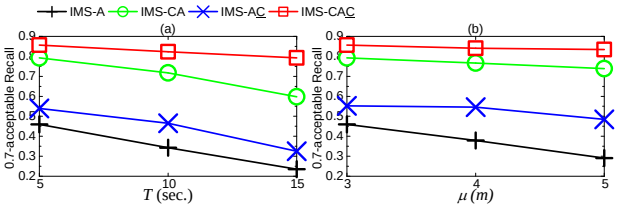

Figure 13: Construction Effectiveness on Synthetic Data

The results reported in different settings of T and μ show that the overall framework IMS-CAC works very effectively at constructing m-semantics even when 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 6.1.4. In the experiments, parameter $\mathcal{T}_s$ 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 $\mathcal{T}$ to 120 min.. The precisions of different methods are reported in Figure 14.

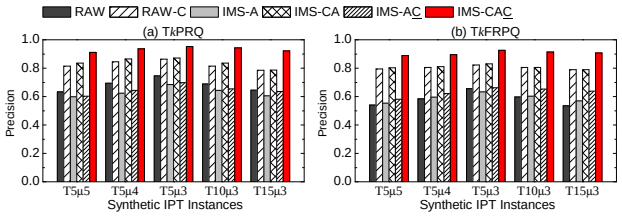

Figure 14: Query Answering Effectiveness on Synthetic Data

Referring to the results of $T5\mu5$, $T5\mu4$ and $T5\mu3$ in Figure 14(a) and (b), precisions of all methods decrease when μ increases in both queries. All methods with data cleaning clearly outperform the others in the tests. When μ increases to $5m$, our proposed IMS-CAC can still have a precision of 91.06% for TkPRQ and 88.84% for TkFRPQ, showing a 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 14(a) and (b), the precision of each method decreases with an increasing T. However, IMS-CAC 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 Modeling. A semantic trajectory is generally defined as a (GPS) data trace enhanced with annotations and/or complementary segmentations [35]. Alvares *et al.* [4] propose to extract stop and move events from trajectory sample points based on the geographic information. Marketos *et al.* [33] design a trajectory reconstruction method to transform raw trajectories into valuable information needed by specific application. Yan *et al.* [46] propose a kind of hybrid trajectory that encapsulates both geometry and semantics of mobility data and supports different levels of abstraction. Su *et al.* [41] 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 human-readable short text.

Our work differs from these works in three aspects. First, our work studies the mobility data collected from indoor space. The limitations of indoor positioning, complex indoor topology, and particular mobility constraints [31] make our problem distinctive from those in free spaces [4, 33] or road networks [46, 41]. Second, our work translates raw positioning data into a number of triplets with generic mobility events, whereas work [41] generates the unstructured texts. Third, our work further complements the ms-sequence by leveraging the mobility knowledge obtained from historical data, which is not considered in works [4, 46, 33]. Consequently, all these works are unsuitable to address our problem.

A recent work [38] mines the indoor stop-by pattern as a sequence of occurence regions from uncertain RFID data. Different input data and problem definition make it unsuitable for our study.

Indoor Mobility Data Cleaning. Indoor positioning data is more discrete and less accurate compared to GPS data due to the limitations of indoor positioning technologies [27]. To clean raw RFID data, Chen *et al.* [10] design a Bayesian inference approach that makes use of duplicate RFID readings and prior knowledge about readers and environment. The likelihood is captured by designing a state detection model. Baba *et al.* clean the RFID tracking data by utilizing either the integrity constraints [7, 6] implied by indoor 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. The method requires only minimal information of reader deployment but it is able to handle both false positives and false negatives.

In these works, RFID data represents object location as symbolic position, whereas our work considers a type of positioning data in which an object's location at a certain time is described as a point. Also, our cleaning method identifies and repairs three types of data errors in one pass, while works [7] and [6] can only handle one specific type of data error in a separate, specialized process. Moreover, our inference method for recovering the missing data is different from the state detection model [10], probabilistic graph [6], and multi-variate HMM [5] in three points. First, our inference is based on the mobility constraints captured at the level of user mobility semantics. Second, we model and capture the similar movements between each pair of indoor semantic regions, while works [10, 6, 5] build the prior knowledge in a global scope. Third, unlike other works, our transition probability computation considers the effect of the corresponding indoor path's walking length.

There also exist cleaning methods for indoor positioning data in the format of points. Filtering methods such as Bayesian filtering [14] and Kalman filtering [47] are commonly used in this data setting. Besides, assuming indoor trajectories that conform with underlying route networks, Prentow *et al.* [37] propose a bootstrapping approach and adjunctive matching algorithms to mitigate the positioning error bias. Differently, our cleaning method utilizes indoor mobility constraints that are not embedded in those filtering based methods [14, 47], and does not need the hypothesis [37] that locations must be constrained in the route networks.

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 splitting method to split the cleaned data sequence into snippets according to the spatial and temporal 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 interesting to integrate with other types of mobility data, such as RFID or Bluetooth tracking data, to make our framework more generic. It is also useful to enrich the mobility semantics by making use of other user behavior data such as transaction logs or check-ins.

9. REFERENCES

[1] InLocation Alliance. http://www.in-location-alliance.com/.

[2] List of countries by smartphone penetration. http://goo.gl/pdtvMM.

[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. In *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] 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.

[11] G. Deak, K. Curran, and J. Condell. A survey of active and passive indoor localisation systems. *Computer Communications*, 35(16): 1939–1954, 2012.

[12] M. Delafontaine, M. Versichele, T. Neutens, and N. Van de Weghe. Analysing spatiotemporal sequences in Bluetooth tracking data. *Applied Geography*, 34: 659–668, 2012.

[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. *IEEE Pervasive Computing*, 2(3): 24–33, 2003.

[15] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory pattern mining. In *KDD*, pp. 330–339, 2007.

[16] A. Globerson, and T. S. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *NIPS*, pp. 553–560, 2008.

[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] I. Hwang and Y. J. Jang. Process mining to discover shoppers pathways at a fashion retail store using a WiFi-base indoor positioning system. *IEEE Trans. Auton. Sci. Eng.*, 14(4): 1786–1792, 2017.

[20] V. ManoChitra, S. Shanthi, and S. S Shajahaan. Mining mobile sequential pattern in a location aware environment. *International Journal of Computer Science and Mobile Computing*, 2(10): 159–165, 2013.

[21] 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.

[22] C. S. Jensen, H. Lu, and B. Yang. Graph model based indoor tracking. In *MDM*, pp. 122–131, 2009.

[23] D. B. Johnson and D.A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile computing*, pp. 153–181. 1996.

[24] B. Kanagal and A. Deshpande. Online filtering, smoothing and probabilistic modeling of streaming data. In *ICDE*, pp. 1160–1169, 2008.

[25] 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*, vol. 11(3): 231, 2001.

[26] 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.

[27] 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.

[28] H. Li, H. Lu, L. Shou, G. Chen, and K. Chen. In search of indoor dense regions: An approach using indoor positioning data. *IEEE Trans. Knowl. Data Eng.*, vol.30, 2018. Print.

[29] H. Lu, X. Cao, and C. S. Jensen. A foundation for efficient indoor distance-aware query processing. In *ICDE*, pp. 438–449, 2012.

[30] 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.

[31] H. Lu, B. Yang, and C. S. Jensen. Spatio-temporal joins on symbolic indoor tracking data. In *ICDE*, pp. 816–827, 2011.

[32] X. Lu, E. Wetter, N. Bharti, A. J. Tatem, and L. Bengtsson. Approaching the limit of predictability in human mobility. *Scientific reports*, vol. 3, 2013.

[33] G. Marketos, E. Frentzos, I. Ntoutsi, N. Pelekis, A. Raffaetà, and Y. Theodoridis. Building real-world trajectory warehouses. In *MobiDE*, pp. 8–15, 2008.

[34] C. Papageorgiou, K. Birkos, T. Dagiuklas, and S. Kotsopoulos. Modeling human mobility in obstacle-constrained ad hoc networks. *Ad hoc networks*, 10(3):421–434, 2012.

[35] C. Parent, S. Spaccapietra, C. Renso, *et al*. Semantic trajectories modeling and analysis. *ACM Comput. Surv.*, 45(4), 2013.

[36] 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.

[37] T S. Prentow, A. Thom, H. Blunck, and J. Vahrenhold. Making sense of trajectory data in indoor spaces. In *MDM*, pp. 116-121, 2015.

[38] S-Y. Teng, W-S. Ku, and K-T. Chuang. Toward Mining Stop-by Behaviors in Indoor Space. *TSAS*, 3(2), 2017.

[39] L. Radaelli, D Sabonis, H. Lu, and C. S. Jensen. Identifying typical movements among indoor objects–concepts and empirical study. In *MDM*, pp. 197–206. 2013.

[40] 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.

[41] 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.

[42] M.F. Worboys. Modeling indoor space. In *SIGGPATIAL*, pp. 1–6, 2011.

[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] B. Yang, H. Lu, and C. S. Jensen. Scalable continuous range monitoring of moving objects in symbolic indoor space. In *CIKM*, pp. 671–680, 2009.

[45] B. Yang, H. Lu, and C. S. Jensen. Probabilistic threshold k nearest neighbor queries over moving objects in symbolic indoor space. In *EDBT*, pp. 335–346, 2010.

[46] Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, and K. Aberer. Semantic trajectories: Mobility data computation and annotation. *TIST*, 4(3):49, 2013.

[47] J. Yim, C. Park, J. Joo, and S. Jeong. Extended Kalman Filter for wireless LAN based indoor positioning. *Decision Support Systems*, 45(4): 960–971, 2008.

[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.