Indoor Mobility Semantics Annotation Using Coupled Conditional Markov Networks

Huan Li[†] Hua Lu[†] Muhammad Aamir Cheema[‡] Lidan Shou[§] Gang Chen[§]

[†]Department of Computer Science, Aalborg University, Denmark
[‡]Faculty of Information Technology, Monash University, Australia
[§]College of Computer Science and Technology, Zhejiang University, China
{lihuan, luhua}@cs.aau.dk, aamir.cheema@monash.edu, {should, cg}@zju.edu.cn

***Abstract*—Indoor mobility semantics analytics can greatly benefit many pertinent applications. Existing semantic annotation methods mainly focus on outdoor space and require extra knowledge such as POI category or human activity regularity. However, these conditions are difficult to meet in indoor venues with relatively small extents but complex topology. This work studies the annotation of indoor *mobility semantics* that describe an object's mobility event (*what*) at a semantic indoor region (*where*) during a time period (*when*). A coupled conditional Markov network (C2MN) is proposed with a set of feature functions carefully designed by incorporating indoor topology and mobility behaviors. C2MN is able to capture probabilistic dependencies among positioning records, semantic regions, and mobility events jointly. Nevertheless, the correlation of regions and events hinders the parameters learning. Therefore, we devise an alternate learning algorithm to enable the parameter learning over correlated variables. The extensive experiments demonstrate that our C2MN-based semantic annotation is efficient and effective on both real and synthetic indoor mobility data.**

I. Introduction

The pervasiveness of indoor localization technologies [7] has been generating unprecedented amounts of mobility data indoors. Analyzing indoor mobility data can reveal interesting findings that are otherwise hard to obtain, e.g., popular indoor locations [13], [14] and frequent indoor patterns [22].

Multiple studies [10], [27], [31] have shown that analyzing semantics can significantly promote knowledge extraction from mobility data. Many existing works on mobility analytics require external data such as texts [10] (e.g., user posts) and contextual geographic information [31] (e.g., home and working place) in semantics extraction. However, such external data is often unavailable, which entails methods able to extract semantics from raw mobility data.

In this paper, we study the semantic annotation problem on raw indoor mobility data. We intend to understand *when-where-what* about user movements. This representation is comprehensive and informative in that it represents spatial, temporal and semantic information jointly [19], [26], [31]. Specifically, given an object's indoor positioning records each consisting of a location l and a timestamp t, we annotate them by a sequence of *mobility semantics*, each of which includes a *semantic region*, a time period, and a *mobility event*. We use *m-semantics* to call such mobility semantics.

Indoor semantic regions are often pre-defined with particular semantics by data analysts. For example, a semantic region can be a cashier or a shop in a mall. In this paper, we assume semantic regions do not overlap. A mobility event refers to some interesting movement pattern. We introduce two generic indoor patterns, namely *stay* and *pass*.[1] A stay indicates that an object has been *staying in* a semantic region for a sufficiently long period of time for a particular purpose that is fulfilled in that region. In contrast, a pass tells that an object has *passed by* a semantic region but there is no particular purpose associated with that pass. We use regions and events to refer to semantic regions and mobility events, respectively.

Figure 1 gives an example of the m-semantics for a tourist in Copenhagen Central Station. Suppose we obtain the tourist's indoor Wi-Fi positioning records as a pair of location and timestamp. Such records are uncertain as the underlying localization is imprecise. Plotting the positioning records on the station's floorplan, we can annotate a sequence of m-semantics. For example, m-semantics (*John's Hotdog Deli*, *12:21:32-12:22:15*, *stay*) means that the tourist stayed in a snack bar *John's Hotdog Deli* during time interval [12:21:32, 12:22:15]. As an indoor region itself usually carries rich semantics determined by its particular usage, combining a stay with the corresponding region is useful to disclose rich information about user behavior. In this example, the tourist is likely to buy some food in that snack bar. Furthermore, the distinction between stay and pass is useful in pertinent scenarios. For example, for the owner of a region *Food Market* to estimate the conversion rate of people who have been in his shop, he needs to know the number of people with *pass* and that with *stay*. In this case, m-semantics like (*Food Market*, *12:42:19-12:44:26*, *pass*) will be necessary.

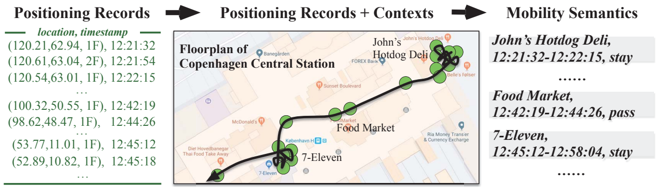
Positioning Records
Positioning Records + Contexts
Mobility Semantics
location, timestamp
(120.21,62.94, 1F), 12:21:32
(120.61,63.04, 2F), 12:21:54
(120.54,63.01, 1F), 12:22:15
...
(100.32,50.55, 1F), 12:42:19
(98.62,48.47, 1F), 12:44:26
...
(53.77,11.01, 1F), 12:45:12
(52.89,10.82, 1F), 12:45:18
...
Floorplan of Copenhagen Central Station
Banegården
FOREX Bank
John's Hotdog Deli
John's Hotdog Deli
Belle's Følser
Sunset Boulevard
McDonald's
FoodMarket
Food Market
København H
Ria Money Transfer & Currency Exchange
Die! Hovedbanegar Thai Food Take Away
7-Eleven
7-Eleven
John's Hotdog Deli, 12:21:32-12:22:15, stay
......
Food Market, 12:42:19-12:44:26, pass
7-Eleven, 12:45:12-12:58:04, stay
......

Fig. 1: From Positioning Records to Mobility Semantics

In general, m-semantics provide an intuitive understanding of object behaviors in the physical world, enabling further semantics-oriented queries and analyses. M-semantics extract

[1] The outdoor patterns that carry similar meanings are known as *stop* and *move* in other literatures [2], [19], [26].

from uncertain and redundant positioning records user behavior related semantics, facilitating multiple downstream applications, e.g., semantic location prediction [31] and activity recommendation [29] for users. As a unified representation, m-semantics is independent of underlying localization techniques that generate different types of mobility data.

Given a sequence of discrete positioning records, we propose to annotate m-semantics by a label-and-merge method as illustrated in Figure 2. In particular, we label each record with a region and an event, and merge the consecutive records having the same region and event labels to form m-semantics. For example, the records from time t_2 to t_{i-1} in Figure 2 can be merged as $(r_D, [t_2, t_{i-1}], stay)$. An advantage of this method is that the merging can be performed at different region granularities to meet various application needs. E.g., in a large mall we can construct m-semantics according to different shops or different business areas.

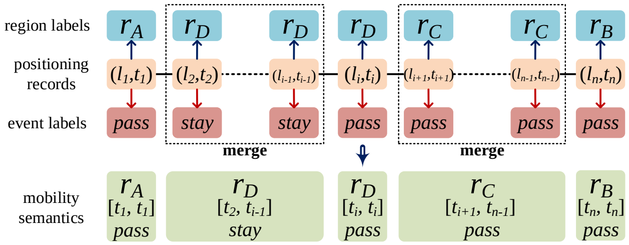

Fig. 2: Label-and-Merge for Annotating Mobility Semantics

However, labeling positioning records with regions and events is still a challenging task in indoor settings: First, positioning records obtained by indoor localization often suffer from various issues such as high positioning errors and low sampling rates[2]. Spatial and temporal uncertainties inherent in the data make it hard to identify an indoor object's exact whereabouts and mobility states. Second, an indoor venue usually has a relatively small extent but complex topology, leading to a compact distribution of regions and complex mobility behaviors under indoor topology. This makes the labeling even more difficult. Existing annotation techniques for outdoors make use of POI category [26] or assumptions such as human activity regularity [24]. However, in indoor spaces the same type of POIs often placed together but object movements are quite random. Third, moving objects' underlying regions and events are always correlated such that an object labeled as stay within a time interval should not appear in multiple regions during that interval. Such correlations over the sequence significantly increase the complexity of labeling.

To address these challenges, we propose a novel graphical model named *coupled conditional Markov network* (C2MN). Specifically, C2MN captures the *joint* relationship among the positioning records, region labels, and event labels by abstracting multiple types of probabilistic dependencies commonly-seen in spatiotemporal sequences. These dependencies embed 1) the correlation between a record and a label at a single time instance, 2) the correlation between two consecutive labels, and 3) the correlation between different types of labels at consecutive time instances. Such probabilistic representation learned from historical labeled sequences helps overcome the spatiotemporal uncertainties. To cope with the unique indoor setting, a set of feature functions are then carefully designed in C2MN to incorporate useful knowledge about indoor topology and indoor mobility behaviors.

Nevertheless, the flexible dependency definition in C2MN complicates the parameter learning as the target labels are coupled across time. To this end, a novel alternate learning paradigm is devised to progressively estimate optimal parameters for one label type with the other label type being fixed.

To sum up, this paper makes the following contributions.

- We formulate the problem of indoor mobility semantics annotation and solve it based on sequence labeling techniques (Section II).
- We design a C2MN model and a set of feature functions to label the semantic regions and mobility events jointly for an indoor positioning sequence (Section III).
- We devise an alternate learning algorithm that takes into account the correlations of region labels and event labels in parameter estimation. (Section IV).
- We conduct extensive experiments on both real and synthetic data to evaluate the efficiency and effectiveness of our C2MN-based annotation method. (Section V)

In addition, Section VI reviews the related work; Section VII concludes the paper and discusses future work.

II. Preliminaries

Table I lists the notations used throughout this paper.

TABLE I: Notations

Symbol	Meaning
$P_o = \langle (l_1,t_1), \ldots, (l_n,t_n) \rangle$	object o's positioning sequence
r	an indoor semantic region
$\tau = [t_s, t_e]$	a time period
$e \in \{stay, pass\}$	an indoor mobility event
$ms = (r, \tau, e)$	an m-semantics
$MS_o = \langle ms_1, \ldots, ms_m \rangle$	object o's m-semantics sequence

A. Problem Definition

In our setting, an indoor positioning system aperiodically reports a positioning record $\theta(l,t)$ for an object o, meaning o was observed at a location l at timestamp t. In most indoor positioning systems [7], l is a triplet (x,y,f), i.e., a 2D point $(x,y) \in \mathbb{R}^2$ on a floor $f \in \mathbb{N}$. Given a time period T, we define an object's *positioning sequence* (p-sequence) as follows.

Definition 1 (Positioning Sequence)**.** *An object o's positioning sequence over time period* T *is a time-ordered sequence of positioning records of o, denoted as* $P_{o,\mathsf{T}} = \langle (l_1,t_1), \ldots, (l_n,t_n) \rangle$ *such that* $[t_1, t_n] \subseteq \mathsf{T}$.

Definition 2 (Mobility Semantics)**.** *An object o's* mobility semantics *is a triplet* $ms(r, \tau, e)$*, where r is a semantic region,* τ *is a time period, and e is a mobility event.*

Essentially, an indoor space can be divided into a number of *indoor partitions* like rooms and hallways by walls and doors.

[2] Unlike GPS, indoor localization mainly relies on wireless technology is susceptible to multiple environmental factors.

We assume each semantic region consists of one or more such partitions. We also assume a mobility event refers to either stay or pass. Nevertheless, more events can be defined by using ontology like *Simple Event Model*[3], and our learning-based model can be extended to handle multiple events by designing specialized cost functions based on the event properties. Due to space limit, we skip such extensions in this paper.

Definition 3 (M-Semantics Sequence)**.** *An object o's* m-semantics sequence *(ms-sequence) over time period* T *is a time-ordered sequence* $MS_{o,\mathsf{T}}$ *of o's m-semantics:* $\forall ms_i, ms_j \in MS_{o,\mathsf{T}}$, $ms_i.\tau \subseteq \mathsf{T}, ms_j.\tau \subseteq \mathsf{T}, ms_i.\tau \cap ms_j.\tau = \varnothing$.

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

Research Problem (M-Semantics Annotation)**.** *Given a p-sequence* $P_o = \langle (l_1,t_1), \ldots, (l_n,t_n) \rangle$*, the goal of* m-semantics annotation *is to generate the most-likely ms-sequence* $MS_o = \langle ms_1, \ldots, ms_m \rangle$ *for* P_o*.*

In this paper, we distinguish labeling and annotation in that labeling is for time instances and annotation is for time periods. As illustrated in Figure 2, our m-semantics are annotated after each positioning record on P_o has been labeled with a region and an event. Next, we give preliminaries of conditional Markov network (CMN) for sequence labeling.

B. Conditional Markov Networks

For typical labeling problems [20], [26], the goal is to configure the optimal *target variable* $\mathbf{Y}$ that maximizes that conditional distribution over the *observation* $\mathbf{X}$. In a CMN [20], also known as conditional random field (CRF), the conditional distribution $\mathsf{P}(\mathbf{y} \mid \mathbf{x})$ is defined by a full set C of *cliques*, each being a fully-connected sub-graph in the network. Specifically, $\mathsf{P}(\mathbf{y} \in \mathbf{Y} \mid \mathbf{x} \in \mathbf{X})$ is factorized into a product of *clique potentials* $\phi_c(\mathbf{x}_c, \mathbf{y}_c)$, where $c \in \mathsf{C}$ is a clique and $\mathbf{x}_c, \mathbf{y}_c$ are the observed nodes and target nodes in the clique c, respectively. Clique potentials are functions that define "compatibility" among clique nodes, i.e., the larger the potential value, the more likely the variable configuration for the clique nodes. To guarantee non-negativeness for optimization, $\phi_c(\mathbf{x}_c, \mathbf{y}_c)$ is often described by log-linear combination of feature functions $\mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_c)$, i.e., $\phi_c(\mathbf{x}_c, \mathbf{y}_c) = \exp\{\mathbf{w}_c^\top \cdot \mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_c)\}$. Consequently, the conditional distribution can be written as

$$\begin{aligned} \mathsf{P}(\mathbf{y} \mid \mathbf{x}) &= \frac{1}{Z(\mathbf{x})} \prod_{c \in \mathsf{C}} \phi_c(\mathbf{x}_c, \mathbf{y}_c) \\ &= \frac{1}{Z(\mathbf{x})} \prod_{c \in \mathsf{C}} \exp\{\mathbf{w}_c^\top \cdot \mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_c)\} \\ &= \frac{1}{Z(\mathbf{x})} \exp\Big\{ \sum_{c \in \mathsf{C}} \mathbf{w}_c^\top \cdot \mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_c) \Big\} \end{aligned} \tag{1}$$

where $Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{c \in \mathsf{C}} \phi_c(\mathbf{x}_c, \mathbf{y}_c)$ is the normalization function.
Unrolled CMNs. To apply CMNs to the sequence data with different lengths, we can unroll the dependencies of variables over the sequence while linking every two consecutive nodes generated for each variable. However, the unrolled net would be rather complex as thousands of nodes will be involved over the time. To make it possible to learn a large number of parameters associated with the probabilistic dependencies, *parameter sharing* [21] is used to enable learning the same parameters for the same *clique template* in a CMN. In particular, a clique template specifies a particular relational structure among a set of nodes. In parameter sharing, each clique template corresponds to one weight vector and the gradient of the weight vector is given by the sum of the gradients computed for all cliques satisfying that template. Using parameter sharing, we rewrite Equation 1 as

$$\mathsf{P}(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \frac{1}{Z(\mathbf{x}, \mathbf{w})} \exp\Big\{ \sum_{ct \in CT} \sum_{c \in \mathsf{C}(ct)} \mathbf{w}_{ct}^\top \cdot \mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_c) \Big\} \tag{2}$$

where $ct \in CT$ is one of the clique templates, $\mathbf{w}_{ct} \subseteq \mathbf{w}$ is the part of weight vector associated with template ct, and $c \in \mathsf{C}(ct)$ denotes a clique satisfying ct. As a result, for all the cliques in $\mathsf{C}(ct)$, we only need to design one feature function and estimate one weight vector. This method significantly reduces the number of parameters to estimate in an unrolled CMN.

III. Model Design for M-Semantics Annotation

Given observation $P = \langle (l_1,t_1) \ldots, (l_n,t_n) \rangle$, our model aims to jointly infer the most-likely region sequence $R = \langle r_1, \ldots, r_n \rangle$ and event sequence $E = \langle e_1, \ldots, e_n \rangle$. Section III-A introduces the generic labeling framework. Section III-B details the specific feature design for m-semantics annotation.

A. Coupled Conditional Markov Networks

We use a coupled conditional Markov network (C2MN) to define probabilistic dependencies among the positioning records in P, regions in R, and events in E. At each timestamp t_i, we have an observed node $\theta_i \langle l_i, t_i \rangle \in P$ and two target nodes, i.e., a region node $r_i \in R$ and an event node $e_i \in E$. The structure is unrolled over time as depicted in Figure 3. To abstract dependencies between observation and target variables, we recognize four categories of cliques as follows.

- **Matching Cliques** measure the fitness of an observed node and a target node at a particular timestamp.
- **Transition Cliques** capture the label smoothness of two consecutive target nodes.
- **Synchronization Cliques** indicate the transitional consistency for two pairs of an observed node and a target node.
- **Segmentation Cliques** measure the comparability of multiple consecutive pairs of an observed node and a target node, in which the other type of target nodes have the same label.

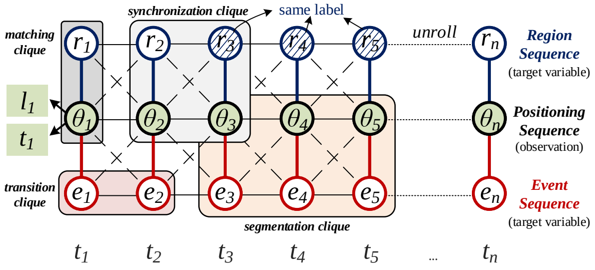

Fig. 3: Coupled Conditional Markov Network for Annotation

[3] https://semanticweb.cs.vu.nl/2009/11/sem/

Example 1. *Referring to Figure 3, a positioning record $\theta_1\langle l_1, t_1\rangle$ and its corresponding region r_1 at time t_1 form a matching clique, so do θ_1 and the corresponding event e_1. Also, two consecutive events e_2 and e_3 form a transition clique. Besides, e_2 and e_3 and their corresponding positioning records θ_2 and θ_3 form a synchronization clique. Finally, suppose regions r_3, r_4, and r_5 have all been labeled as stay, then the corresponding positioning records θ_3, θ_4, and θ_5 and events e_3, e_4, and e_5 form a segmentation clique.*

In general, transition cliques are defined for a target variable (either R or E), matching cliques and synchronization cliques are defined between the observation and a target variable, and segmentation cliques combine the observation with both target variables. Unlike the others, segmentation cliques can only be identified when one target variable has been configured. Learning model parameters with such segmentation cliques will be discussed in Section IV.

C2MN differs from existing sequential learning model in the following aspects. First, it uses undirected edges to connect consecutive nodes of a variable, allowing for a more comprehensive consideration of temporal correlations in a sequence. In contrast, directed graph models such as Hidden Markov Models (HMMs) and Dynamic Bayesian Networks (DBNs) can only define unidirectional dependencies for a variable over the time. Second, it uses transition cliques and synchronization cliques to capture temporal correlations for target (hidden) nodes, while sequential models like linear-chain CRF cannot model dependencies for hidden nodes. Third, it not only considers the association between variables in the node-level, but also captures the coupling of target variables in the label-level by using segmentation cliques. This is an important feature of C2MN compared to other CMN-like models.

Our C2MN-based framework only describes potential dependencies among different types of nodes without specific feature function design. Next, we will consider the unique characteristics of indoor topology and mobility behaviors, and incorporate them into the feature functions specified for annotating indoor mobility data.

B. Feature Function Design for M-Semantics Annotation

As two target variables (R and E) are introduced in C2MN, we divide the dependencies captured by the clique templates (see Section II-B) into the *region relevant dependencies* related to the region nodes in R, and *event relevant dependencies* involving the event nodes in E. Table II lists the corresponding feature functions, which are explained as follows.

(1) Spatial Matching Function $\mathbf{f}_{sm}(\theta_i, r_i)$ measures the probability of matching a region r_i given the observed location $\theta_i.l$. Due to the indoor positioning errors, the possible location of an object observed at $\theta_i.l$ can be represented as an uncertainty region $UR(\theta_i.l, v)$ modeled as a circular region centered at $\theta_i.l$ with a radius v. The larger the intersection area of $UR(\theta_i.l, v)$ and a region r_i, the higher the likelihood of matching $\theta_i.l$ to the region r_i. The function is thus defined as

$$\mathbf{f}_{sm}(\theta_i, r_i) = \frac{UR(\theta_i.l, v) \cap Area(r_i)}{UR(\theta_i.l, v)} \tag{3}$$

where $Area(r_i)$ is the covering area of r_i. Referring to a location estimate l_1's uncertainty region illustrated in Figure 4, we know that both r_A and r_B are possible labels of l_1, but r_B has a higher probability in $\mathbf{f}_{sm}$. Equation 3 only considers pure spatial relationship. In some buildings, the regions used more frequently can have a higher probability of matching. Possibly, an alternative design is to include the normalized historical region frequency as a multiplier to the right-hand side of Equation 3.

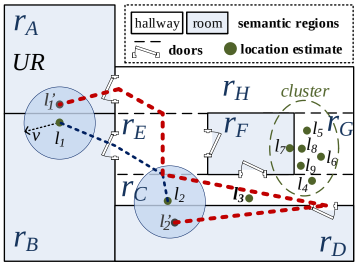

Fig. 4: An Example Indoor Floorplan

(2) Event Matching Function $\mathbf{f}_{em}(\theta_i, e_i)$ measures the correlation between positioning record θ_i and event e_i. In particular, the spatiotemporal features corresponding to θ_i should conform to the corresponding mobility event, i.e., stay or pass. In our observation, positioning records associated with a stay always have their location estimates and timestamps packed together. Therefore, a density-based clustering algorithm over the location and time attributes of positioning records is useful to distinguish stay and pass. To be specific, the records being clustered as core and border points are more likely to be associated with a stay, and the possibility of core points should be higher than that of border points. In contrast, the noise points are closer to a pass. Referring to Figure 4, consecutive reports l_4 to l_9 in a cluster are likely to correspond to a stay while the sparsely distributed ones l_1, l_2, and l_3 are likely to associate with pass. Formally, the function is given as

$$\mathbf{f}_{em}(\theta_i, e_i) = \begin{cases} 1, & \text{if } (e_i, \theta_i.\mathrm{D}) = (\mathit{stay}, \mathrm{core}) \text{ or } (\mathit{pass}, \mathrm{noise}); \\ \alpha, & \text{if } (e_i, \theta_i.\mathrm{D}) = (\mathit{stay}, \mathrm{border}); \\ \beta, & \text{if } (e_i, \theta_i.\mathrm{D}) = (\mathit{pass}, \mathrm{border}); \\ 0, & \text{otherwise.} \end{cases}$$

where $0 < \beta < \alpha < 1$ are two constant values and $\theta_i.\mathrm{D} \in \{\mathrm{core, border, noise}\}$ indicates θ_i's spatiotemporal density when clustering the whole p-sequence. We employ the st-DBSCAN algorithm [3] that requires three parameters: 1) ε_s is a distance threshold for location attributes; 2) ε_t is a distance threshold for time attributes; 3) pt_m is a number threshold. A cluster is formed only if it contains at least pt_m data instances and any two instances in it are within the spatial distance ε_s and temporal distance ε_t from each other. We use st-DBSCAN instead of the original DBSCAN because time attribute should be considered to cluster *consecutive* records.

(3) Space Transition Function $\mathbf{f}_{st}(r_i, r_{i+1})$ measures the coherence between two consecutive region labels. Referring to Figure 4, for consecutive locations l_2 and l_3 having a certain elapsed time, they should more probably be labeled as $\langle r_C, r_C\rangle$ or $\langle r_D, r_D\rangle$ than being labeled in different regions

TABLE II: Feature Functions for Indoor M-Semantics Annotation

Clique Catagory	Region Relevant Dependencies	Event Relevant Dependencies
Matching Cliques	(1) *Spatial Matching Function* $\mathbf{f}_{sm}(\theta_i, r_i)$	(2) *Event Matching Function* $\mathbf{f}_{em}(\theta_i, e_i)$
Transition Cliques	(3) *Space Transition Function* $\mathbf{f}_{st}(r_i, r_{i+1})$	(4) *Event Transition Function* $\mathbf{f}_{et}(e_i, e_{i+1})$
Synchronization Cliques	(5) *Spatial Consistency Function* $\mathbf{f}_{sc}(\theta_i, \theta_{i+1}, r_i, r_{i+1})$	(6) *Event Consistency Function* $\mathbf{f}_{ec}(\theta_i, \theta_{i+1}, e_i, e_{i+1})$
Segmentation Cliques	(7) *Event-based Segmentation Function* $\mathbf{f}_{es}(c_{es}^{i:j})$	(8) *Space-based Segmentation Function* $\mathbf{f}_{ss}(c_{ss}^{i:j})$

(e.g., $\langle r_D, r_C \rangle$). We use the distance between indoor regions to reflect the extent of label change. In particular, we define the transition cost from r_i to r_{i+1} as the expectation of *minimum indoor walking distance* (MIWD) [17] from a point p in r_i to a point q in r_{i+1}. The larger the distance, the greater the label change. As a result, the function is given as

$$\mathbf{f}_{st}(r_i, r_{i+1}) = \exp\{ -\gamma_{st} \cdot \mathsf{E}_{p \in r_i, q \in r_{i+1}}[\mathsf{d}_I(p,q)] \} \tag{4}$$

where $\mathsf{d}_I()$ computes MIWD and γ_{st} is a scale parameter within $(0,1)$. In the above example, $\mathbf{f}_{st}(r_C, r_C)$ (i.e., 1) is greater than $\mathbf{f}_{st}(r_D, r_C)$. Another possible design is to add the effect of elapsed time to region transition cost, i.e., the longer the elapsed time, the lower the impact of MIWD distance in transition cost. This can be done by including a time-decaying multiplier $e^{-\gamma' \cdot (t_{i+1} - t_i)}$ to Equation 4 where γ' is within $(0,1)$.

(4) Event Transition Function $\mathbf{f}_{et}(e_i, e_{i+1})$ measures the smoothness between two consecutive events. Similar to space transition function, we use the difference of the two event labels to reflect the smoothness and define the function as

$$\mathbf{f}_{et}(e_i, e_{i+1}) = \begin{cases} 1, & \text{if } e_i = e_{i+1}; \\ 0, & \text{otherwise.} \end{cases}$$

(5) Spatial Consistency Function $\mathbf{f}_{sc}(\theta_i, \theta_{i+1}, r_i, r_{i+1})$ measures the consistency between the distance at the region level (r_i to r_{i+1}) and that at the raw location level ($\theta_i.l$ to $\theta_{i+1}.l$). In our observation, wrong region mapping on the indoor map will result in a more complex indoor path compared to that of the reasonable region mapping. The penalty for such a complex path can be measured by its difference to the Euclidean distance between the origin and destination. The more complex the indoor path, the greater the difference. Referring to Figure 4, possibly we can label l_1's and l_2's regions as r_A and r_D and in this case the red dashed line can be a representative indoor path from some possible location l_1' in r_A to some possible location l_2' in r_D. Such a path is significantly more complex than the straight-line path between the locations, which indicates that the underlying region labels r_A and r_D could be abnormal. Instead, if we label l_1's and l_2's regions as $\langle r_B, r_C \rangle$, then their indoor path (blue dotted) is simpler. Compared to $\langle r_A, r_D \rangle$, we think that $\langle r_B, r_C \rangle$ is closer to the real situation. The function is defined based on the difference between the two distances at different levels.

$$\mathbf{f}_{sc}(\theta_i, \theta_{i+1}, r_i, r_{i+1}) = \exp\{ -|\mathsf{E}_{p \in r_i, q \in r_{i+1}}[\mathsf{d}_I(p,q)] - \mathsf{d}_E(\theta_i.l, \theta_{i+1}.l)| \} \tag{5}$$

The time decaying effect can also be considered for spatial consistency. Similar to extending Equation 4, a multiplier $e^{-\gamma'' \cdot (t_{i+1} - t_i)}$ can be optionally used in Equation 5.

(6) Event Consistency Function $\mathbf{f}_{ec}(\theta_i, \theta_{i+1}, e_i, e_{i+1})$ measures the consistency of the moving speed between observations θ_i, θ_{i+1} and their underlying events e_i, e_{i+1}. The faster the moving speed, the more likely the corresponding labels are pass than stay. Given the moving speed revealed by observations, we measure its consistency with the number of pass in the event labels. The feature function is defined as

$$\mathbf{f}_{ec}(\theta_i, \theta_{i+1}, e_i, e_{i+1}) = \exp\{ -\big|\min(1, \gamma_{ec} \cdot \tfrac{\theta_{i+1}.l - \theta_i.l}{\theta_{i+1}.t - \theta_i.t}) - \tfrac{\mathsf{I}_{\triangleright}(e_i) + \mathsf{I}_{\triangleright}(e_{i+1})}{2}\big| \}$$

where $\mathsf{I}_{\triangleright}(e_i)$ is an indicator function which equals 1 if $e_i =$ *pass* or 0 otherwise, and γ_{ec} is scale parameter for the moving speed computed as $\frac{\theta_{i+1}.l - \theta_i.l}{\theta_{i+1}.t - \theta_i.t}$. When the speed is high, the closer the two events are to pass, the higher the consistency is, and vice versa. For example, when the speed is 0, the function gets the maximum value 1 when both θ_i and θ_{i+1} are stay.

(7) Event-based Segmentation Function $\mathbf{f}_{es}(c_{es}^{i:j})$ measures the compatibility among the clique nodes that have the same event label. Formally, an event-based segmentation $c_{es}^{i:j} = \{\theta_i, \ldots, \theta_j, r_i, \ldots, r_j\}$ is formed if $\forall e_x, e_y, i \le x \le y \le j, e_x = e_y$ and $e_x \ne e_z$ for $z = j+1$ or $z = i-1$. Provided that the clique nodes correspond to a certain event label, the key features extracted from the positioning records and region labels should match the event label as much as possible. We define the feature function as

$$\mathbf{f}_{es}(c_{es}^{i:j}) = \big(2 \cdot \mathsf{I}_{\triangleright}(c_{es}^{i:j}.e) - 1\big) \cdot \begin{pmatrix} \mathit{DISTNUM}(r_i, \ldots, r_j) \\ \sum_{x=i}^{j-1} \mathsf{d}_E(\theta_x.l, \theta_{x+1}.l)/(\theta_j.t - \theta_i.t) \\ -\mathit{TURNNUM}(\theta_i.l, \ldots, \theta_j.l) \end{pmatrix}^{\mathsf{T}}$$

where the vector in the right returns three real values extracted from the nodes in $c_{es}^{i:j}$, namely the distinct number of region labels over $\{r_i, \ldots, r_j\}$, moving speed between θ_i and θ_j, and number of turns[4] between θ_i and θ_j. In our observation, a stay tends to correspond to *fewer* regions, *lower* moving speed and *larger* number of turns, whereas a pass should be the opposite. Therefore, we use $2 \cdot \mathsf{I}_{\triangleright}(c_{es}^{i:j}.e) - 1$ to indicate whether the feature function is positively or negatively correlated with the underlying event $c_{es}^{i:j}.e$. Referring to Figure 4, if the reports l_4 to l_9 have all been labeled as stay, then we expect that the number of their corresponding region labels is as small as possible, to be consistent with a stay event. This helps label some noise points (e.g., l_4 in r_C and l_7 in r_F) in the clique.

(8) Space-based Segmentation Function $\mathbf{f}_{ss}(c_{ss}^{i:j})$ measures the compatibility among the clique nodes that have the same region label. Formally, a space-based segmentation $c_{ss}^{i:j} = \{\theta_i, \ldots, \theta_j, e_i, \ldots, e_j\}$ is formed if $\forall r_x, r_y, i \le x \le y \le j, r_x = r_y$ and $r_x \ne r_z$ for $z = j+1$ or $z = i-1$. Similar to the event-based segmentation function, we expect the features revealed

[4] For a location $\theta_i.l$, if the angle between the line from $\theta_{i-1}.l$ to $\theta_i.l$ and the line from $\theta_i.l$ to $\theta_{i+1}.l$ exceeds 90 degrees, it is considered to be a turn.

from $c_{ss}^{i:j}$ to be close to the truth of moving inside the corresponding region. The feature function is defined as

$$\mathbf{f}_{ss}(c_{ss}^{i:j}) = \begin{pmatrix} -NUM(e_i,\dots,e_j)/(\theta_j.t-\theta_i.t) \\ -(\sum_{x=i}^{j-1}\mathbf{f}_{et}(e_x,e_{x+1})/(\theta_j.t-\theta_i.t) \\ \mathsf{I}_{\triangleright}(e_i)+\mathsf{I}_{\triangleright}(e_j) \end{pmatrix}^{\mathsf{T}}$$

where the first two features refer to the number and the event transition number per second over the sequence $\langle e_i,\dots,e_j\rangle$, and the last feature means the number of stay for the first and last record in that segmentation. Intuitively, the mobility state does not change very frequently over a period of time inside the same region, so there should have smaller event number and event transition number in such a clique. Moreover, given that the region of the current segmentation is different from the segmentation before and after, the first and last event labels in this segmentation are more likely to be pass events. After extraction, feature values in $\mathbf{f}_{es}$ and $\mathbf{f}_{ss}$ need to be normalized.

In our design, each function evaluates the labeling plausibility in one aspect, and the C2MN framework chooses the label configuration with the highest *overall* evaluation as the final result. The importance of each function in the framework is determined by the weights learned from training data.

IV. Supervised Learning of C2MN

A. Objective Function

Given fully labeled data in the form of (P,R,E), we need to estimate the weights $\mathbf{w}_{ct}\subseteq\mathbf{w}$ associated with the feature functions for each clique template (see Equation 1). To find the optimal weights $\mathbf{w}$ that maximize the conditional distribution $\mathsf{P}(R,E\mid P,\mathbf{w})$, we rewrite Equation 1 into *negative log-likelihood* form, plus a regularization term to avoid overfitting. The regularization term is a zero-mean Guassian prior with constant variance σ on each component of $\mathbf{w}$. Consequently, we have the following **objective function** for C2MN.

$$\begin{aligned}\mathsf{L}(\mathbf{w}) &= -\log\mathsf{P}(R,E\mid P,\mathbf{w})+\frac{\mathbf{w}^{\mathsf{T}}\mathbf{w}}{2\sigma^2}\\ &= \sum_{ct\in CT}\sum_{c\in\mathsf{C}(ct)}\Big(-\mathbf{w}_{ct}^{\mathsf{T}}\cdot\mathbf{f}_c(P_c,R_c,E_c)\Big)+\log Z(P,\mathbf{w})+\frac{\mathbf{w}^{\mathsf{T}}\mathbf{w}}{2\sigma^2}\\ &= \sum_{ct\in CT}\Big(-\mathbf{w}_{ct}^{\mathsf{T}}\cdot\sum_{c\in\mathsf{C}(ct)}\mathbf{f}_c(P_c,R_c,E_c)\Big)+\log Z(P,\mathbf{w})+\frac{\mathbf{w}^{\mathsf{T}}\mathbf{w}}{2\sigma^2}\\ &= \sum_{ct\in CT}-\mathbf{w}_{ct}^{\mathsf{T}}\cdot\mathbf{f}_{ct}(P_{ct},R_{ct},E_{ct})+\log Z(P,\mathbf{w})+\frac{\mathbf{w}^{\mathsf{T}}\mathbf{w}}{2\sigma^2}\\ &= -\mathbf{w}^{\mathsf{T}}\cdot\mathbf{f}(P,R,E)+\log Z(P,\mathbf{w})+\frac{\mathbf{w}^{\mathsf{T}}\mathbf{w}}{2\sigma^2}\end{aligned}$$

where function $\mathbf{f}_{ct}(P_{ct},R_{ct},E_{ct})=\sum_{c\in\mathsf{C}(ct)}\mathbf{f}_c(P_c,R_c,E_c)$ gives a summation over the extracted features for all cliques satisfying the template ct, and vector $\mathbf{w}$ and feature vector given by $\mathbf{f}(P,R,E)$ are the stacking of $\mathbf{w}_{ct}$ and $\mathbf{f}_{ct}$ over all clique templates, respectively. It can be shown that $\mathsf{L}(\mathbf{w})$ is convex relative to $\mathbf{w}$ and has a global optimum that can be searched using numerical gradient algorithms such as quasi-Newton method [16]. However, the global optimization of $\mathsf{L}(\mathbf{w})$ has two major challenges. On the one hand, computing $Z(P,\mathbf{w})$ needs to consider all possible label configurations for unknown variables R and E, which requires an expensive inference procedure at each iteration. On the other hand, R and E are correlated, which further complicates the label configuration especially when we involve segmentation cliques in.

To reduce the nodes to consider for computing expected feature values, we assume that each target node is only related to its immediate neighbors, i.e., the nodes in its *Markov blanket* [20]. Although this assumption simplifies the dependencies of random variables, it captures the relationship between non-neighboring nodes through iterative learning upon the whole network. Based on the assumption, we are able to optimize the *pseudo-likelihood* $\mathsf{PL}(\mathbf{w})$ instead of the global likelihood $\mathsf{L}(\mathbf{w})$ that involves all variable nodes, i.e.,

$$\begin{aligned}\mathsf{PL}(\mathbf{w}) &= -\log\sum_{y_i\in R\cup E}\mathsf{P}(y_i\mid\mathrm{MB}(y_i),\mathbf{w})+\frac{\mathbf{w}^{\mathsf{T}}\mathbf{w}}{2\sigma^2}\\ &= -\sum_{y_i\in R\cup E}\big(\mathbf{w}^{\mathsf{T}}\cdot\mathbf{f}(y_i,\mathrm{MB}(y_i))+\log Z(\mathrm{MB}(y_i),\mathbf{w})\big)+\frac{\mathbf{w}^{\mathsf{T}}\mathbf{w}}{2\sigma^2}\end{aligned} \tag{6}$$

The above pseudo-likelihood is computed as the sum of all local likelihoods $\mathsf{P}(y_i\mid\mathrm{MB}(y_i))$ plus the prior term, where y_i is a target node that can be a region $r\in R$ or an event $e\in E$, and $\mathrm{MB}(y_i)$ is the Markov blanket of y_i. Accordingly, the gradient of $\mathsf{PL}(\mathbf{w})$ is given as

$$\nabla\mathsf{PL}(\mathbf{w})=\sum_{y_i\in R\cup E}\Big(-\mathbf{f}(y_i,\mathrm{MB}(y_i))+\mathsf{E}_{\mathsf{P}(y_i'\mid\mathrm{MB}(y_i),\mathbf{w})}[\mathbf{f}(y_i',\mathrm{MB}(y_i),\mathbf{w})]\Big)+\frac{\mathbf{w}}{\sigma^2} \tag{7}$$

where $\mathsf{E}_{\mathsf{P}(y_i'\mid\mathrm{MB}(y_i),\mathbf{w})}[\mathbf{f}(y_i',\mathrm{MB}(y_i),\mathbf{w})]$ is the *expected feature values* over the distribution $\mathsf{P}(y_i'\mid\mathrm{MB}(y_i),\mathbf{w})$. Therefore, $\nabla\mathsf{PL}(\mathbf{w})$ can be regarded as the difference between the *empirical feature values* given by $\mathbf{f}(y_i,\mathrm{MB}(y_i))$ and the expected feature values over the possible labels of the region and event nodes in $\mathrm{MB}(y_i)$, plus the prior term.

As a result, the gradient in Equation 7 can be computed more efficiently based on those local expectations with respect to $\mathbf{f}(y_i',\mathrm{MB}(y_i),\mathbf{w})$ than based on a global expectation with respect to all possible region sequences and event sequences. Still, the empirical feature values $\mathbf{f}(y_i,\mathrm{MB}(y_i))$ are hard to know if y_i is a node in a segmentation clique. This is because a segmentation clique can only be identified if R or E has been configured. In Section IV-B, we introduce an alternate learning paradigm to enable an effective way to learn parameters when involving segmentation cliques.

B. Alternate Learning with MCMC Inference

We iteratively update the weights $\mathbf{w}$ using a quasi-Newton method L-BFGS [16], which requires evaluating both objective value $\mathsf{PL}(\mathbf{w})$ and its gradient $\nabla\mathsf{PL}(\mathbf{w})$. The idea of alternate learning is to configure one type of target nodes and update weights for another type of target nodes alternately. Specifically, in each step, we infer one target variable (say A, A can be either R or E) using the weights from the previous step, and compute $\mathsf{PL}(\mathbf{w})$ and $\nabla\mathsf{PL}(\mathbf{w})$ to update the weights for the other target variable (say B). The order of A and B is then exchanged in the next step. This alternating evaluation stops until the weights associated with both A and B converge.

Next, we introduce how to compute $\mathsf{PL}(\mathbf{w})$ and $\nabla\mathsf{PL}(\mathbf{w})$ with one target variable A fixed. For computing $\mathsf{PL}(\mathbf{w})$ in Equation 6, the normalization function $Z(\mathrm{MB}(y_i),\mathbf{w})$ requires a costly summation over all possible label configurations. For the ease of computation, we use an MCMC (Markov Chain Monte Carlo) inference to approximate the objective value. Suppose we have already obtained a weight vector $\hat{\mathbf{w}}$ and a configured variable $\bar{A}$ from the previous steps[5]. Then we can obtain M random samples of $\bar{B}^{(j)} = \langle \bar{b}_1^{(j)}, \ldots, \bar{b}_n^{(j)} \rangle$ $(1 \leq j \leq M)$ by MCMC sampling over the distribution $\mathsf{P}(b_i \mid \mathrm{MB}(b_i, \bar{A}), \hat{\mathbf{w}})$. Particularly, b_i $(1 \leq i \leq n)$ is a target node of B, and $\mathrm{MB}(b_i, \bar{A})$ is the Markov blanket of b_i given the currently configured sequence $\bar{A}$. Consequently, $\mathsf{PL}(\mathbf{w})$ at the current step can be approximated as

$$\begin{aligned}\mathsf{PL}(\mathbf{w}) \approx\ & \mathsf{PL}(\hat{\mathbf{w}}) + \sum_{b_i \in B} \Big(\log\{ \frac{1}{M} \sum_{j=1}^{M} \exp\{ (\mathbf{w} - \hat{\mathbf{w}})^{\mathsf{T}} \cdot \Delta\bar{\mathbf{f}}^{(j)}(b_i, \bar{A}) \} \} \Big) \\ & + \frac{\mathbf{w}^{\mathsf{T}}\mathbf{w} - \hat{\mathbf{w}}^{\mathsf{T}}\hat{\mathbf{w}}}{2\sigma^2}\end{aligned} \tag{8}$$

where $\Delta\bar{\mathbf{f}}^{(j)}(b_i, \bar{A}) = \mathbf{f}(\mathrm{MB}(b_i, \bar{A}), \bar{b}_i^{(j)}) - \mathbf{f}(\mathrm{MB}(b_i, \bar{A}), b_i)$ is the difference between the *sampled* feature values using $\hat{\mathbf{w}}$ and $\bar{A}$ and the empirical feature values of training data. Equation 8 only estimates value of $\mathsf{PL}(\mathbf{w})$ relative to $\mathsf{PL}(\hat{\mathbf{w}})$, whereas it is still useful in searching the optimal weights since the best approximation is met when $\hat{\mathbf{w}}$ is close to the optimal $\mathbf{w}$.

For evaluating $\nabla\mathsf{PL}(\bar{\mathbf{w}})$ in Equation 7, we need to conduct an MCMC inference again to get M new samples of $B^{(j)} = \langle b_1^{(j)}, \ldots, b_n^{(j)} \rangle$ using the *updated* weights $\mathbf{w}$ instead of $\hat{\mathbf{w}}$ from the previous steps. Accordingly, Equation 7 is adapted as

$$\nabla\mathsf{PL}(\mathbf{w}) \approx \sum_{b_i \in B} \frac{1}{M} \sum_{j=1}^{M} \Delta\mathbf{f}^{(j)}(b_i, \bar{A}) + \frac{\mathbf{w}}{\sigma^2} \tag{9}$$

where $\Delta\mathbf{f}^{(j)}(b_i, \bar{A}) = \mathbf{f}(\mathrm{MB}(b_i, \bar{A}), b_i^{(j)}) - \mathbf{f}(\mathrm{MB}(b_i, \bar{A}), b_i)$ is the difference between sampled feature values using **new weights** $\mathbf{w}$ and $\bar{A}$ and empirical feature values of training data.

C. Parameter Learning Algorithm

The learning algorithm is presented in Algorithm 1, which receives a random weight vector $\mathbf{w}_0$ and searches for the optimal weights $\mathbf{w}$. In the beginning, we need to first initialize one variable, either R or E. Here, we choose to configure E since it only has two possible labels and its initialization can be quickly done by applying st-DBSCAN [3] on the p-sequences.[6] In particular, all noise points after clustering are regarded as pass and others are regarded as stay. As a result, we obtain a configured sequence $\bar{E}$ for each object (line 1). Next, the algorithm initializes $\hat{\mathbf{w}}$ to store the weights that achieve the best PL so far (line 2). Afterwards, it calls function *Alternate_Learn* to estimate new weights using $\mathbf{w}_0$ and the variables $\bar{E}$ configured for all objects (line 3).

In each step, function *Alternate_Learn* receives the weights $\mathbf{w}$ and the configured variable $\bar{A}$ of all objects. For each object, it samples M sequences $\bar{B}^{(j)}$ and computes the feature value difference $\Delta\bar{\mathbf{f}}^{(j)}(b_i, \bar{A})$ (lines 5–8). After the sampling, it computes $\nabla\mathsf{PL}(\mathbf{w})$ according to Equation 9 (line 9). It then computes $\mathsf{PL}(\mathbf{w})$ (lines 10–16). If the function is called for the first time (i.e., there is no update for $\hat{\mathbf{w}}$ so far), it directly assigns $\Delta\mathbf{f}^{(j)}(b_i, \bar{A})$ to $\Delta\bar{\mathbf{f}}^{(j)}(b_i, \bar{A})$ (lines 10-11). Otherwise, it computes $\mathsf{PL}(\mathbf{w})$ according to Equation 8 (line 13). If the new $\mathsf{PL}(\mathbf{w})$ is better than $\mathsf{PL}(\hat{\mathbf{w}})$ (line 14), $\hat{\mathbf{w}}$ and the corresponding $\Delta\bar{\mathbf{f}}^{(j)}(b_i, \bar{A})$ are both updated with $\mathbf{w}$ (lines 15–16). In lines 11 and 16, reusing feature value difference sampled by the best $\mathbf{w}$ improves the learning efficiency.

Once the new estimates $\nabla\mathsf{PL}(\mathbf{w})$ and $\mathsf{PL}(\mathbf{w})$ are obtained, the algorithm runs L-BFGS algorithm to generate new weights $\bar{\mathbf{w}}$ (line 17). If the *Chebyshev* distance (i.e., the maximum element-wise distance) between $\bar{\mathbf{w}}$ and $\mathbf{w}$ is smaller than a given threshold δ (line 18), $\bar{\mathbf{w}}$ is returned as the optimum because a convergence is met (line 19). Otherwise, the algorithm updates $\mathbf{w}$ with $\bar{\mathbf{w}}$ (line 21) and decides which variable should be fixed for the next step (lines 22–26). If the partial weights $\bar{\mathbf{w}}_A$ associated with A have been convergent (line 22), the next calling of *Alternate_Learn* will be executed with the previous $\bar{A}$ (line 23). Otherwise, the calling will be executed with $\bar{B}$ that is obtained by averaging the M samples $\bar{B}^{(j)}$ obtained at the current step (lines 24–26). A maximum iteration number *max_iter* is used as the stopping criterion of Algorithm 1.

Algorithm 1: Alternate Learning with MCMC Inference

```
   Input: inital weights $\mathbf{w}_0$
   Output: optimal weights $\mathbf{w}$
   /* use $E$ as the first-configured variable    */
1  run st-DBSCAN to generate sequence $\bar{E}$ for each object;
2  $\hat{\mathbf{w}} = \mathbf{w}_0$; $\mathsf{PL}(\hat{\mathbf{w}}) = \mathsf{PL}(\mathbf{w}) = 0$;
   /* at most max_iter iterative calling          */
3  Alternate_Learn($\mathbf{w}_0$, $\bar{E}$) on all objects;
4  Function Alternate_Learn ($\mathbf{w}$, $\bar{A}$)
5      for each object do
6          for $j$ = 1 to $M$ do
7              run MCMC with $\mathbf{w}$ and $\bar{A}$ to generate $\bar{B}^{(j)}$;
8              compute feature value difference $\Delta\mathbf{f}^{(j)}(b_i, \bar{A})$;
9          compute $\nabla\mathsf{PL}(\mathbf{w})$ using Equation 9;
10         if first time calling then
11             $\Delta\bar{\mathbf{f}}^{(j)}(b_i, \bar{A}) = \Delta\mathbf{f}^{(j)}(b_i, \bar{A})$ for $1 \leq i \leq n$, $1 \leq j \leq M$;
12         else
13             compute $\mathsf{PL}(\mathbf{w})$ using Equation 8;
14             if $\mathsf{PL}(\mathbf{w}) < \mathsf{PL}(\hat{\mathbf{w}})$ then
15                 $\mathsf{PL}(\hat{\mathbf{w}}) = \mathsf{PL}(\mathbf{w})$; $\hat{\mathbf{w}} = \mathbf{w}$;
16                 $\Delta\bar{\mathbf{f}}^{(j)}(b_i, \bar{A}) = \Delta\mathbf{f}^{(j)}(b_i, \bar{A})$ for $1 \leq i \leq n$, $1 \leq j \leq M$;
17     run L-BFGS with $\mathsf{PL}(\mathbf{w})$, $\nabla\mathsf{PL}(\mathbf{w})$ to get new weights $\bar{\mathbf{w}}$;
18     if $||\bar{\mathbf{w}} - \mathbf{w}||_\infty \leq \delta$ then
19         return $\bar{\mathbf{w}}$;
20     else
21         $\mathbf{w} = \bar{\mathbf{w}}$;
22         if $||\bar{\mathbf{w}}_A - \mathbf{w}_A||_\infty \leq \delta$ then
23             Alternate_Learn($\mathbf{w}$, $\bar{A}$);
24         else
25             get $\bar{B}$ by averaging $\bar{B}^{(j)}$ for $1 \leq j \leq M$ for each object;
26             Alternate_Learn($\mathbf{w}$, $\bar{B}$) on all objects;
```

V. Experimental Studies

C2MN is implemented using CRF++ [1], an open-source implementation of CRFs in C++. All experiments are done with a Xeon 10-core 2.20GHz CPU + 128GB memory server.

[5] If here A refers to R, then B refers to E, and vice versa.

[6] An alternative way is to first configure R by applying the nearest-neighbor region matching to the p-sequences. Using different first-configured variable (E or R) will be experimentally studied in Section V-B3.

A. Compared Methods and Performance Metrics

Methods in Comparison. We introduce several alternatives to our method denoted as C2MN as follows.

- SMoT [2] uses a speed threshold to distinguish stay and pass events on a sequence, and the nearest-neighbor regions as region labels for the representative locations in an event.
- HMM+DC uses an HMM for region labeling in which semantic regions are hidden states and positioning records (distributed to corresponding grids) are observations. Parameters are estimated via frequency counting and regions are inferred by Viterbi decoding. For event labeling, an st-DBSCAN Clustering (abbreviated as DC) is used in which the core and border points are regarded as stay and noise points as pass. The method was previously applied to our indoor trajectory translation system TRIPS [12].
- SAP (*Semantic annotation platform* [26]) is the state-of-the-art layered framework for trajectory annotation. It first divides stay (stop) and pass (move) segments according to some segmentation algorithm. Next, it labels each stay segment with a region using an HMM, in which the observation probability between the segment (its locations) and a region is modeled as their intersection ratio over the Gaussian density distribution of the locations; it labels each location in the pass segment with its nearest region. We selected the dynamic-velocity-based and density-area-based algorithms for segmentation (see [26]), and the corresponding overall methods are denoted as SAPDV and SAPDA, respectively.
- CMN decouples the region and event variables by removing the two types of segmentation cliques from C2MN. Assuming the two variables are independent, CMN infers region sequence and event sequence asynchronously.
- We also introduce four variants that removes parts of dependencies in C2MN, namely C2MN/Tran without transition cliques, C2MN/Syn without synchronization cliques, C2MN/ES without event-based segmentation cliques, and C2MN/SS without space-based segmentation cliques.

All methods were provided with the same labeled data but they work with data differently: HMM+DC and CMN label regions and events in two independent, asynchronous ways; SMoT, SAPDV, and SAPDA annotate events and regions in two sequential steps. These five methods cannot model the correlation of regions and events such that their two annotation procedures cannot be integrated but are *separated*. In contrast, all C2MN-based methods infer regions and events *jointly*.

Performance Metrics. We define *labeling accuracy* as the fraction of positioning records that receive correct labels. As each record has two labels, we implement **region accuracy** (RA) and **event accuracy** (EA) that measure accuracy in terms of region labels and event labels, respectively. We then define **combined accuracy** (CA) that combines RA and EA by a tradeoff parameter λ such that $CA = \lambda \cdot RA + (1-\lambda) \cdot EA$. Usually, RA's requirement is stricter than EA's since an event label makes no sense if the region label is wrong. On the contrary, analysts can still know a user's whereabouts with a correct region label. In the evaluation, we use a large $\lambda = 0.7$ for CA. We also define **perfect accuracy** (PA) as the fraction of records having *both* region and event labels correct.

B. Experiments on Real Data

1) Settings: **Dataset.** We collected real data from a Wi-Fi positioning system in a seven-floor shopping mall in Hangzhou, China from Jan 1 to Jan 31, 2017. The average daily number of devices (i.e., MAC addresses) and positioning records were around 7,647 and 2,907,904, respectively. Since a device may leave the mall that causes a long-time-interval discontinuity in its p-sequence, we performed data preprocessing as follows: i) We divided a p-sequence with large time intervals into multiple p-sequences. In particular, if the time difference of two consecutive records θ_i, θ_{i+1} exceeds a threshold η, we regarded θ_i as the end of the current p-sequence and θ_{i+1} as the start of a new p-sequence. ii) We filtered out the p-sequences with the duration *not* exceeding a threshold ψ. In our experiments, we set η to 3 minutes and ψ to 30 minutes. Consequently, we obtained 5,218,361 positioning records for 44,863 p-sequences. The characteristics of the final dataset are summarized in Table III.

TABLE III: Statistics of Real Dataset

average number of records per sequence	116.32
average duration per sequence	2227.9 sec.
positioning data error based on MIWD	2 ∼ 25 meters
average sampling rate	∼ 1/15 Hz

Indoor Space. Based on the decomposition algorithm in [25], we divided the mall space into 3,742 regular indoor partitions and obtained 6,534 (virtual) doors that connect these partitions. 202 shops in the mall were selected as semantic regions according to application needs, each consisting of a number of partitions. To facilitate spatial computations in feature extraction, we used an accessibility base graph [17] to maintain indoor topology and an R-tree to index all partitions and their corresponding semantic regions. Their total size is 12.6 MB. The shortest indoor distances between doors were pre-computed to speed up computations on MIWD, which resulted in an additional 990.8 MB memory consumption.

Model Training. We used the Event Editor[7] in TRIPS [12] to annotate the positioning sequences rendered on the indoor map as we were unable to know a device's exact whereabouts. In particular, we asked two reviewers familiar with the mall space to go through the m-semantics suggested by a computer-aided tool of Event Editor, adjust the time range for an event, and edit the corresponding region if they think it is wrong. The trajectory visualization helped identify those obvious positioning errors and the reviewers' double-checking helped eliminate ambiguity. For the visually annotated sequences, we used 10-fold cross-validation with a 70/30 train/test split. Though third-party annotated data cannot ensure 100% validity, the cross-validation on large datasets can reflect a model's ability to learn the human-annotated data for semantic annotation. The performance in different split settings will be reported in Section V-B2. We used a Gaussian prior $\sigma^2 = 0.5$ for pseudo-likelihood in Equation 6 and pseudo-likelihood all converged

[7] More details of the Event Editor are available at longaspire.github.io/trips/

in experiments. We set $\delta = 1e^{-3}$ for the convergence criterion of Algorithm 1, the maximum training iteration $max\_iter = 90$, and the MCMC instance number per step $M = 800$. We applied st-DBSCAN with parameters ($\varepsilon_s = 8$m, $\varepsilon_t = 60$s, $pt_m = 4$) to configure the inital E in Algorithm 1. We tuned $v = 15m$ in $\mathbf{f}_{sm}$, $\alpha = 0.8$, $\beta = 0.6$ in $\mathbf{f}_{em}$, $\gamma_{st} = 0.1$, and $\gamma_{ec} = 0.2$ for the best evaluation performance of C2MN. Labeling a p-sequence with around 100 positioning records takes less than 600ms, acceptable even for online services.

2) Labeling Accuracy: **Comparison of Different Methods.** Table IV reports the labeling accuracy for different methods listed in Section V-A. First, C2MN performs best in all measures. Its region accuracy *RA* is around 0.95 and event accuracy *EA* is higher than 0.96. Moreover, over 88.6% of C2MN's labeled records have both their regions and events correct, as indicated by the perfect accuracy *PA*. Next, we compare the *best achieved* results of SMoT, HMM+DC, SAPDV, SAPDA, and CMN. Among them, SMoT is the worst in both region and event labeling. HMM+DC outperforms SMoT especially for a significant lift in *EA*. The advantage of the density-based method over speed-based method in our setting can also be verified by *EA*s of SAPDA and SAPDV. In our problem, the location uncertainty and complex indoor topology formed by rooms and doors altogether make it imprecise to compute the speed between consecutive records. Compared to finding a distinguishable speed threshold, the density-based methods consider the spatiotemporal distribution from a global view and therefore achieve better segmentation results.

TABLE IV: Results of Labeling Accuracy

Methods	*RA*	*EA*	*CA*	*PA*
SMoT	0.7254	0.8125	0.7515	0.6687
HMM+DC	0.7443	0.8769	0.7841	0.6780
SAPDV	0.7028	0.8296	0.7408	0.6485
SAPDA	0.7394	0.8781	0.7810	0.6943
CMN	0.8860	0.8983	0.8897	0.6684
C2MN/Tran	0.8994	0.9109	0.9026	0.7474
C2MN/Syn	0.9332	0.9073	0.9254	0.8268
C2MN/ES	0.9222	0.9495	0.9304	0.7387
C2MN/SS	0.9014	0.9525	0.9167	0.7616
C2MN	**0.9492**	**0.9691**	**0.9552**	**0.8866**

As the segmentation results affect subsequent region annotations, SAPDA is also higher than SAPDV on *RA*. HMM+DC and SAPDA perform almost equally. They both use HMMs, but HMM+DC is for record level while SAPDA is for stay segments only. SAPDA has higher *EA* but lower *RA* than HMM+DC. Through analysis we find that SAPDA's lower *RA* is mainly because it uses the nearest regions to annotate pass segments. CMN is better than all above methods as it better captures probabilistic dependencies among nodes. However, *PA* of CMN is slightly lower than the others, probably because CMN cannot ensure the coupling of region and event labels.

As introduced in Section V-A, the two-way methods (CMN and HMM+DC) and two-step methods (SMoT, SAPDV, and SAPDA) cannot learn the interaction between regions and events from labeled data and use them for labeling. This leads to a labeling accuracy lower than our C2MN method.

Finally, we compare C2MN to its variants trained in the same setting. Specifically, C2MN/Tran performs the worst among all C2MN structures, showing that the transition dependencies have a significant impact on sequential models. C2MN/Syn achieves the highest *RA* but lowest *EA* among all variants. This shows that the synchronization cliques removed from C2MN/Syn have a minor effect on region labeling but are useful for event labeling. Compared to C2MN, *RA* of C2MN/ES and *EA* of C2MN/SS both decrease. Moreover, their *PA*s are relatively low, even much lower than C2MN/Syn that still retains two types of segmentation cliques.

In general, those separated annotation methods perform poorly, and C2MN with a complete structure outperforms all its variants. The results show that the joint labeling on regions and events can significantly improve the overall accuracy.

Effect of Training Data Fraction. We vary the fraction of training data from 40% to 80%, and report *CA* and *PA* in Figures 5 and 6, respectively. With more training p-sequences, both measures of each method increase moderately. When the ratio of training data increases to 70%, the improvement tends to be flat as the parameter learning becomes saturate. A slight increase at this time mainly comes from the reduction of testing data. Besides, C2MN/ES and C2MN/SS without segmentation cliques stabilize more rapidly, implying that they can learn no more from larger training datasets.

Effect of MCMC Instances. We vary the number *M* of MCMC instances per step from 400 to 1000. The *RA* and *EA* of different C2MN-based methods are shown in Figures 7 and 8, respectively. As *M* increases, more sequences are sampled in each step, which improves the parameter learning efficiency. In Figure 7, *RA*s of most methods remain stable when *M* is up to 800. This shows that *M*=800 can well approximate the distribution of region variable by inference over observations and the current learned parameters. Differently, most methods' *EA*s reported in Figure 8 change slightly for different *M* values. As the event variable has only two label values, an *M* over 400 is large enough for its MCMC inference.

3) Training Efficiency: This section studies the effect of different model parameters on the training time cost.

Comparison of Different Structures. We report the training time of different C2MN-based methods in Figure 9 in different *max_iter* settings. Since CMN trains the region and event parts separately, we only report the longest training time of the two parts to make a fair comparison. CMN takes the least time in all tests because it does not consider the complex parameter learning for segmentation cliques. This greatly reduces the overall computational cost. Likewise, the costs of C2MN/ES and C2MN/SS are also clearly lower than other methods. C2MN has the highest costs in most *max_iter* settings. However, it can still finish the training within 4 hours when *max_iter* increases at 120. Considering its improvement in labeling accuracy, such an offline training cost is acceptable.

Effect of Training Data Fraction. Figure 10 reports the training time with different fractions of training data. With more training p-sequences, the parameter learning needs to consider the features extracted from more positioning records. Thus, the training time for each method increases accordingly. Nevertheless, as parameter sharing is introduced in C2MN that

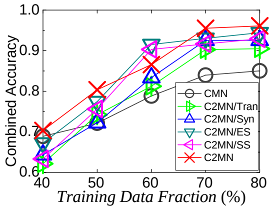

Fig. 5: *CA* vs. % of Training Data

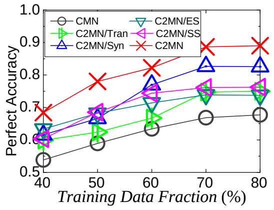

Fig. 6: *PA* vs. % of Training Data

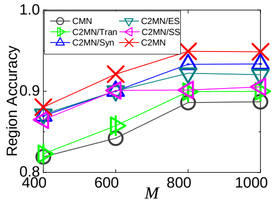

Fig. 7: *RA* vs. *M*

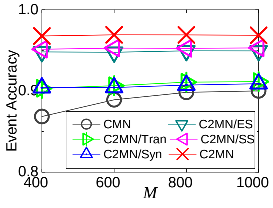

Fig. 8: *EA* vs. *M*

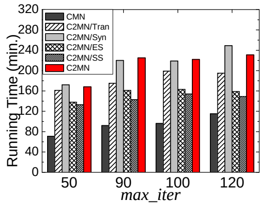

Fig. 9: Training Time vs. *max_iter*

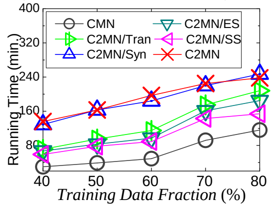

Fig. 10: Training Data Fraction

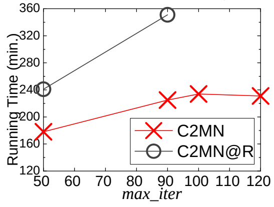

Fig. 11: First-Configured Variable

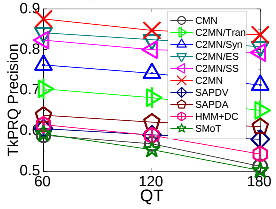

Fig. 12: Precision for T*k*PRQ

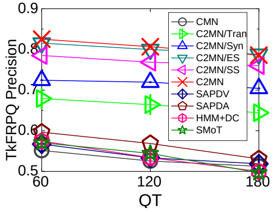

Fig. 13: Precision for T*k*FRPQ

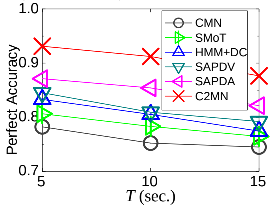

Fig. 14: *PA* vs. *T*

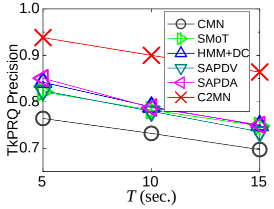

Fig. 15: T*k*PRQ Precision vs. *T*

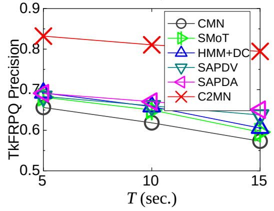

Fig. 16: T*k*FRPQ Precision vs. *T*

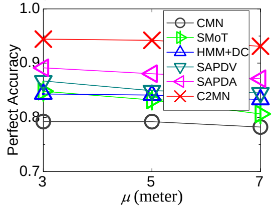

Fig. 17: *PA* vs. *μ*

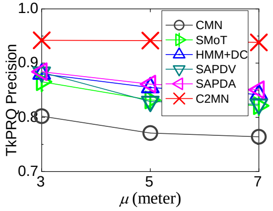

Fig. 18: T*k*PRQ Precision vs. *μ*

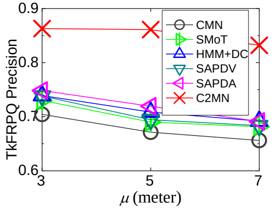

Fig. 19: T*k*FRPQ Precision vs. *μ*

can aggregate the same type of feature vectors at each step, C2MN's training is efficient. When 80% of training data is used, C2MN can still converge within 4 hours.

Effect of First-Configured Variable. As mentioned in Section IV-C, we study the difference of choosing E or R as the first-configured variable in the alternate learning. Differing from the original C2MN, we denote the variant that first configures R by nearest-neighbor matching as C2MN@R. Clearly, C2MN has less training time in different *max_iter* settings whereas C2MN@R's cost increases rapidly. When *max_iter* $= 90$, the training takes nearly 6 hours to finish. Considering the two models work equally well, in our problem setting we suggest using E as the first-configured variable.

4) Quality of M-Semantics in Query Answering: Section V-B2 evaluates the accuracy of labels at the record level. However, the final m-semantics are obtained by merging those labeling results. Therefore, the quality of m-semantics cannot be directly measured by the labeling accuracy. Consider two labeled region sequences $\langle r_A, r_B, r_A\rangle$ and $\langle r_A, r_A, r_B\rangle$ both having the labeling accuracy $2/3$ with respect to the ground truth $\langle r_A, r_A, r_A\rangle$. The former forms three m-semantics, whereas the latter forms only two assuming the corresponding events are labeled the same. For practical use, the latter is of better quality because it is closer to the ground truth. Therefore, we measure the quality of annotated m-semantics in terms of their performance in answering typical queries. Given a query set Q of indoor semantic regions, we introduce two top-k queries.

1) A *Top-k Popular Region Query* (T*k*PRQ) finds k regions from Q that have the most number of *visits*[8].
2) A *Top-k Frequent Region Pair Query* (T*k*FRPQ) finds k most frequent pairs of regions from Q × Q that both have been visited by the same object.

T*k*PRQ and T*k*FRPQ are useful in studies like popular location discovery [13], [14] and frequent pattern mining [22].

We compare the query results of different methods' m-semantics with that computed from the ground truth m-semantics described in Section V-B1. In particular, we use *precision* to measure the ratio of true top-k regions (or region pairs) in the returned top-k results. We issue 10 random queries for each query type and measured the average precision. We fix $k = 60$ and randomly picked 101 (50% of all) semantic regions to the query set Q. We test the queries within a time interval QT varied as 60, 120, 180 minutes from one day.

As shown in Figures 12 and 13, for both types of queries, the precision of all methods decreases with an increasing QT. When a longer QT is used, more relevant data should be considered in the query processing, which involves more data errors and makes the results less effective. Nevertheless, most C2MN-based methods decrease very slowly. Still, the two-way annotation methods (HMM+DC and CMN) and two-step annotation methods (SMoT, SAPDV, and SAPDA) perform poor for both queries. When QT increases to 180 minutes, C2MN can achieve a precision 83.6% for T*k*PRQ and 78.6% for T*k*FRPQ. This shows that the m-semantics annotated by C2MN is of high quality for semantics-relevant queries.

C. Experiments on Synthetic Data

We used synthetic data to further verify the performance of our method when different levels of temporal sparsity and positioning errors are presented in the mobility data. We used the indoor simulator *Vita* [11] to generate a ten-floor building environment with 4 staircases, 1,410 partitions and 2,200 doors. A total of 423 semantic regions were decided upon the partitions at random. We generated 10K moving objects for a period of 4 hours, each having a lifespan varied from 10

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

seconds to 4 hours. Object maximum speed was set to $1.7m/s$ and object movements followed the waypoint model [9]. In particular, each region is considered as a destination, an object moves towards its destination along a pre-planned path. It stays at the destination for a random period from 1 second to 30 minutes after arrival, and moves to the next randomly-decided destination. We recorded an object's location and region every second as the ground truth, and generated its true event labels according to the simulated behavior, i.e., staying at (moving towards) a destination was regarded as a stay (pass) event.

The synthetic datasets are generated according to the ground truth trajectories as follows. After an object has reported an estimate, 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 records of an object. A location estimate is randomly within μ meters from the true location. False floor values and location outliers are added to the reports 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 within 2.5μ-10μ meters from the true location. The *positioning error factor* μ controls the average distance between the positioning location and its true location. To test the effect of temporal sparsity and positioning error, we varied T and μ, respectively. The generated datasets as listed in Table V.

TABLE V: Synthetic Mobility Datasets

Datasets	Parameter Setting	# of Generated Records
$T5\mu3$	$T = 5s, \mu = 3m$	15,230,759
$T5\mu5$	$T = 5s, \mu = 5m$	15,291,495
$T5\mu7$	$T = 5s, \mu = 7m$	15,238,702
$T10\mu7$	$T = 10s, \mu = 7m$	7,695,623
$T15\mu7$	$T = 15s, \mu = 7m$	4,525,429

The total memory costs for accessibility graph, partition R-tree, and shortest door-to-door paths are 470MB. For training parameters, we set $\sigma^2 = 0.2$, $max\_iter = 50$, $M = 500$, $v = 10m$ in $\mathbf{f}_{sm}$, and the others are the same as the counterparts in the experiments on real data. We investigate the perfect accuracy and query precision for alternatives in Section V-A and our method C2MN. All are tuned to the best performance.

Effect of T. Figure 14 reports *PA* for different T values with μ fixed to 7 meters. When varying T from 5s to 15s, i.e., the observed data becomes sparser (see Table V), all methods' *PA*s decrease but C2MN's decreases in the slowest pace. When T=15s, C2MN can still have a *PA* of 0.88. In contrast, the *PA*s of other five methods are never higher than 0.9. CMN performs the worst, which we attribute to the lack of correlations defined for region and event labels.

Given a TkPRQ with $k = 60$, query region set size $|\mathsf{Q}| = 212$, and $\mathsf{TQ} = 120$ minutes, we measure the precision of constructed m-semantics for different Ts. Referring to Figure 15, the TkPRQ precision of each method decreases with an increasing T. However, C2MN decreases slightly, while others deteriorate more rapidly. For a TkFRPQ with $k = 60$, query region set size $|\mathsf{Q}| = 25$, and $\mathsf{TQ} = 120$ minutes, the precision in Figure 16 shows a similar trend with the TkPRQ precision but has lower measures due to a larger ranking space. The results show that our C2MN-based method is very useful for improving the constructed m-semantics, especially when the raw data is temporally sparse.

Effect of μ. We also fixed T to $5s$ and tested with different μs. Referring to *PA* reported in Figure 17, μ only has a slight effect on the measures of each method except SMoT and SAPDV. This is because those two speed-based methods are more susceptible to positioning errors. Still, C2MN always outperforms the other methods clearly, and its *PA* is always higher than 0.92. With different μs, Figures 18 and 19 report the precision for TkPRQ and TkFRPQ (the same query settings as above), respectively. Similar to the results of *PA*, C2MN performs the best in both queries. SAPDA is slightly better than HMM+DC, and SMoT and CMN are the worst. These results demonstrate that our C2MN-based approach works very effectively at constructing m-semantics even when the mobility data quality is low.

VI. Related Work

Semantic Trajectory Representation. Parent et al. [19] define semantic trajectory as a (GPS) data trace enhanced with annotations and/or complementary segmentations. Güting et al. [6] generalize this concept to symbolic trajectory, a sequence of pairs of a time interval and a label referring to any particular term pre-defined by user semantics. Zheng et al. [29] describe a trajectory only by some small regions where moving objects stop for a relatively long time. Such representative regions are called stay points. Nogueira et al. [18] propose a framework with ontology to enrich GPS traces with Linked Open Data (LOD). Compared to these works, the mobility semantics proposed in this paper provide a unified where-when-what view of general user behaviors. Such a structured representation facilitates mobility analytics applications like semantic location prediction [31] or activity recommendation [29].

Semantic Annotation. Giannotti et al. [5] define T-pattern as an ROI sequence with temporal annotations. Zhang et al. [30] derive fined-grained sequential patterns by a top-down splitting of the patterns obtained by POI grouping. Alvares et al. [2] extract stop and move events from trajectory points based on geographical information. Cao et al. [4] propose techniques for extracting semantically meaningful geographical locations visited by users from GPS data. Teng et al. [22] identify indoor stop-by pattern as a sequence of occurrence regions from uncertain RFID data. Different from the sequential patterns [5], [30] and visiting location patterns [2], [4], [22], our work annotates two generic mobility patterns stay and pass, which are flexible for analyzing user behaviors by combining relevant information from semantic regions.

Liao et al. [15] extract activity types and significant places from a person's GPS traces using a hierarchical CRF. Yan et al. [26] propose an HMM-based annotation method to infer stops and POI category for raw GPS records. Wu et al. [24] annotate location records with keywords extracted from geo-referenced social media data by kernel density estimation. By analyzing spatiotemporal regularity, Wu et al. [23] study the personalized annotation that enriches personal GPS records with POI category. Our work differs from these works in several aspects. First, our work searches for a particular region as spatial annotation while other works focus on inferring textual [24] or categorical [15], [23], [26] information for location records. Second, our work annotates both regions and events by considering their mutual association, whereas works [23], [24] are limited to the inference of spatial information. Third, our work uses the characteristics of positioning sequence under indoor topology to capture hidden dependencies among positioning and semantic nodes, while other works demand additional knowledge in geographic spaces, e.g., human activity regularity [15], [23] and POI category [26]. However, such priors are difficult to meet in indoor spaces with relatively small extents but complex topology.

Semantic annotation has been widely studied in Named Entity Recognition (NER). Recently, the BiLSTM-CRF architecture [8] has become de facto standard in which Bi(directional) LSTM encodes the sequence context and CRF decodes the most-likely named entity tags. Compared with NER techniques that use distributed word representations as input, our proposed model handles positioning records directly such that timestamps and uncertain locations must be considered in feature design. By feeding positioning records to the BiLSTM-CRF, its nonlinear transformation and intricate feature learning capabilities may be utilized to ease mobility feature design.

Semantics-rich Spatiotemporal Data Mining. Extracting knowledge from *semantics-rich spatiotemporal data* [28] (e.g., geo-tagged posts) has attracted great research attention recently. Zhang et al. [31] propose an urban activity model that jointly embeds spatial, temporal, and textual units from geo-tagged social media data based on cross-modal representation learning. Aiming at identifying users' mobility behaviors from geo-tagged tweets, Yuan et al. [27] propose a probabilistic model that considers the factors of user, geographic information, time, and activity. By time series analysis of geo-tagged tweets from localized regions, Krumm et al. [10] propose techniques for extracting local events as something that happens at some specific time and place. These works discover user mobility knowledge based on mobility data enhanced with semantics such as POI category, texts, and tweets. In contrast, our work builds a generic, semantic representation of user mobility using positioning records only. Our model works well for indoor venues where semantics-rich spatiotemporal data are usually hard to acquire.

VII. Conclusion And Future Work

This work studies the annotation of indoor mobility data with a semantic region, a time period, and a mobility event. To infer optimal sequences for regions and events, a C2MN is proposed to capture probabilistic dependencies among positioning records, regions, and events. Next, a set of feature functions are designed to incorporate indoor topology and mobility behaviors. Finally, an alternate learning paradigm is proposed to enable parameter estimation over the coupling of regions and events. The experiments verify that our method is efficient and effective, and our method's resultant m-semantics lead to precise answers for typical queries.

For future work, it is useful to define more diverse mobility events for annotation. It is also useful to adapt C2MN to outdoor scenarios, especially when mobility data is sparsely sampled. Moreover, it is interesting to explore how to apply NER techniques (e.g., BiLSTM-CRF) to mobility annotation.

Acknowledgement. This work was supported by the Independent Research Fund Denmark (No. 8022-00366B), the Australian Research Council Grant (Nos. FT180100140 and DP180103411), the NSF-China (No. 61672455), and the NSF of Zhejiang Province of China (No. LY18F020005).

References

[1] CRF++. https://taku910.github.io/crfpp/. Retrieved September, 2019.

[2] L.O. Alvares, V. Bogorny, B. Kuijpers, J. de Macedo, B. Moelans, and A. Vaisman. A model for enriching trajectories with semantic geographical information. *Advances in geographic information systems*, 22, 2007.

[3] D. Birant and A. Kut. ST-DBSCAN: An algorithm for clustering spatial–temporal data. *Data & Knowledge Engineering*, 60(1): 208–221, 2007.

[4] X. Cao, G. Cong, and C. S. Jensen. Mining significant semantic locations from GPS data. *PVLDB*, 3(1): 1009–1020, 2010.

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

[6] R H. Güting, F. Valdés, and M L. Damiani. Symbolic trajectories. *ACM Transactions on Spatial Algorithms and Systems*, 1(2): 7, 2015.

[7] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8): 57–66, 2001.

[8] Z. Huang, W. Xu, and K. Yu. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

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

[10] J. Krumm and E. Horvitz. Eyewitness: Identifying local events via space-time signals in twitter feeds. In *SIGSPATIAL*, pp. 20, 2015.

[11] 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.

[12] H. Li, H. Lu, F. Shi, G. Chen, K. Chen, and L. Shou. TRIPS: A system for translating raw indoor positioning data into visual mobility semantics. *PVLDB*, 11(12): 1918–1921, 2018.

[13] H. Li, H. Lu, L. Shou, G. Chen, and K. Chen. Finding most popular indoor semantic locations using uncertain mobility data. *IEEE Transactions on Knowledge and Data Engineering*, 31(11): 2108–2123, 2018.

[14] H. Li, H. Lu, L. Shou, G. Chen, and K. Chen. In search of indoor dense regions: An approach using indoor positioning data. *IEEE Transactions on Knowledge and Data Engineering*, 30(8): 1481–1495, 2018.

[15] L. Liao, D. Fox, and H. Kautz. Location-based activity recognition. In *NIPS*, pp. 787–794, 2006.

[16] D C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3): 503–528, 1989.

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

[18] T.P. Nogueira, R.B. Braga, C.T. de Oliveira, and H. Martin. FrameSTEP: A framework for annotating semantic trajectories based on episodes. *Expert Systems with Applications*, 92: 533–545, 2018.

[19] C. Parent, S. Spaccapietra, C. Renso, *et al.* Semantic trajectories modeling and analysis. *ACM Computing Surveys*, 45(4), 2013.

[20] C. Sutton, A. McCallum, and K. Rohanimanesh. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research*, 8: 693–723. 2007.

[21] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *UAI*, pp. 485–492, 2002.

[22] S-Y. Teng, W-S. Ku, and K-T. Chuang. Toward mining stop-by behaviors in indoor space. *ACM Transactions on Spatial Algorithms and Systems*, 3(2), 2017.

[23] F. Wu and Z. Li. Where did you go: Personalized annotation of mobility records. In *CIKM*, pp. 589–598, 2016.

[24] F. Wu, Z. Li, W-C. Lee, H. Wang, and Z. Huang. Semantic annotation of mobility data using social media. In *WWW*, pp. 1253–1263, 2015.

[25] X. Xie, H. Lu, and T. B. Pedersen. Efficient distance-aware query evaluation on indoor moving objects. In *ICDE*, pp. 434–445, 2013.

[26] Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, and K. Aberer. Semantic trajectories: Mobility data computation and annotation. *ACM Transactions on Intelligent Systems and Technology*, 4(3): 49, 2013.

[27] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann. Who, where, when and what: discover spatio-temporal topics for twitter users. In *KDD*, 605–613, 2013.

[28] Q. Yuan, C. Zhang, and J. Han. A Survey on spatiotemporal and semantic data mining. *Trends in Spatial Analysis and Modelling*, 43–57, 2018.

[29] K. Zheng and X. Xie. Learning travel recommendations from user-generated GPS traces. *ACM Transactions on Intelligent Systems and Technology*, 2(1): 2, 2011.

[30] C. Zhang, J. Han, L. Shou, J. Lu, and T. La Porta. Splitter: Mining fine-grained sequential patterns in semantic trajectories. *PVLDB*, 7(9): 769–780, 2014.

[31] C. Zhang, K. Zhang, Y. Quan, H. Peng, Y. Zheng, T. Hanratty, S. Wang, and J. Han. Regions, periods, activities: Uncovering urban dynamics via cross-modal representation learning. In *WWW*, pp. 361–370, 2017.