Effective Similarity Search on Indoor Moving-Object Trajectories

Peiquan Jin[1(✉)], Tong Cui[1], Qian Wang[1], and Christian S. Jensen[2]

[1] University of Science and Technology of China, Hefei, China
jpq@ustc.edu.cn
[2] Department of Computer Science, Aalborg University, Aalborg, Denmark

Abstract. In this paper, we propose a new approach to measuring the similarity among indoor moving-object trajectories. Particularly, we propose to measure indoor trajectory similarity based on spatial similarity and semantic pattern similarity. For spatial similarity, we propose to detect the critical points in trajectories and then use them to determine spatial similarity. This approach can lower the computational costs of similarity search. Moreover, it helps achieve a more effective measure of spatial similarity because it removes noisy points. For semantic pattern similarity, we propose to construct a hierarchical semantic pattern to capture the semantics of trajectories. This method makes it possible to capture the implicit semantic similarity among different semantic labels of locations, and enables more meaningful measures of semantic similarity among indoor trajectories. We conduct experiments on indoor trajectories, comparing our proposal with several popular methods. The results suggest that our proposal is effective and represents an improvement over existing methods.

Keywords: Indoor space · Similarity search · Trajectory

1 Introduction

Most people spend most of their time in indoor spaces, such as in homes, office buildings, shopping malls, and airports. The increasingly deployment of indoor positioning technologies offers the possibility to obtain the locations and trajectories of people in indoor spaces [4]. Therefore, there are increasing needs for analyzing the trajectories of indoor moving objects. For instance, it is helpful to find potential shopping patterns by identifying similar paths of customers in a shopping mall [15]. This development calls for the research on similarity search in indoor spaces.

Similarity search is an important issue in many applications. A key objective is to define efficient methods to measure the trajectory similarity for indoor moving objects. Previous efforts on trajectory similarity search focus on outdoor spaces, e.g., Euclidean and road-network spaces [7–10]. However, indoor spaces have unique features compared with outdoor spaces. First, in indoor spaces, we cannot use GPS for positioning. Instead, indoor locations have to be determined through a deployment graph of readers like RFID readers or Wi-Fi adapters [4]. Due to the small-area property of indoor spaces, indoor positions reported by readers are much closer than outdoor locations, yielding a polyline with dense points and introducing many noisy points in trajectory

S.B. Navathe et al. (Eds.): DASFAA 2016, Part II, LNCS 9643, pp. 181–197, 2016.
DOI: 10.1007/978-3-319-32049-6_12

similarity measurement. Second, a typical indoor space consists of different floors and many indoor elements like rooms, doors, hallways, stairs, and elevators. Thus, indoor spaces are actually constrained three-dimensional spaces. Therefore, we need new techniques to compute indoor distances when measuring trajectory similarity.

In this paper, we propose a new approach to measuring trajectory similarity for indoor moving objects. We consider both spatial and semantic similarity between trajectories, and we make the following contributions:

(1) We propose to detect *critical points* in a trajectory and then define spatial similarity based on critical points (Sect. 3.1).
(2) We propose to use a hierarchical categorization of indoor locations to capture the semantic patterns of trajectories, and we define hierarchical semantic similarity for indoor moving objects (Sect. 3.2).
(3) We compare our proposal with the *LCSS*-based and *Edit-Distance*-based methods using synthetic indoor trajectory data. The results suggest that our proposal is effective and improves on existing proposals (Sect. 4).

2 Problem Statement

2.1 Indoor Space

Many models of indoor space have been proposed [1–4]. In this paper, we define indoor space by the following symbolic model

Definition 1 (Indoor Space). An indoor space is represented as a triple:

$$IndoorSpace = (Cell, Sensor, Deployment)$$

Here, *Cell* is a set of cells in the indoor space. According to previous researches, an indoor space can be partitioned into cells [3]. *Sensor* is a set of positioning sensors deployed in the indoor space. Typical sensors are RFID readers, Bluetooth detectors, and Wi-Fi signal receivers. *Deployment* records the placement information of the sensors in the indoor space. □

In real applications, rooms can be regarded as cells and sensors are usually used to identify cells in indoor space, e.g., to identify shops in a shopping mall. Thus, the *Deployment* information of sensors can be pre-determined and maintained in a database.

In order to introduce semantics into the model, we assign semantic labels to each sensor. As a result, the set *Sensor* can be represented as follows.

$$Sensor = \{s | s = (sensorID, location, label)\}$$

The *label* of a sensor provides descriptions on thematic attributes of the location identified by *sensorID*. For instance, we can use labels like "*elevator*" and "*stair*" to indicate the functions of the cell identified by a sensor. We can also use other labels like "*Starbucks*" and "*Burger King*" to annotate semantic features of the cell. The location

of a sensor is a three-dimensional coordinate (x, y, z), which reflects the relative position of the sensor inside the indoor space where the sensor is deployed. In real applications, indoor maps are usually designed by AutoCAD [22]. Thus, if we import an indoor map into a database system, we can simply use the coordinates and floors in the map to represent the locations of sensors.

2.2 Similarity Search in Indoor Spaces

In outdoor spaces, a trajectory is a series of GPS locations, while a trajectory in indoor spaces is typically a series of sensor identifiers. Thus, we first define the indoor location of a moving object as follows:

Definition 2 (Indoor Location). An indoor location *LOC* of a moving object *mo* is defined as LOC_{mo}:

$$LOC_{mo} = \{p | p = (s, [t_s, t_e]) \wedge (t_s < t_e) \wedge s \in Sensor\},$$

Here, s is a sensor, *mo* is the object identifier, t_s is the instant that the object enters the sensor's range and t_e is the instant that the object leaves the sensor range. □

Due to the special properties of some kinds of sensors, an indoor moving object may be detected by two or more sensors simultaneously. Thus, we need to find out the most exact location from a set of sensors' signals. This is an important issue in indoor location-based applications, which has been studied in many previous works on indoor localization [23, 24]. We assume that an indoor moving object has only one indoor location at each time instant. Then, we define the indoor trajectory for a moving object.

Definition 3 (Indoor Trajectory). An indoor trajectory *TR* of a moving object *mo* is defined as a sequence of indoor locations of *mo*:

$$TR_{mo} = \{\langle p_1, p_2, \ldots p_n, \rangle \mid (\forall p_i)(p_i \in LOC_{mo} \wedge (p_i.t_e < p_{i+1}.t_s))\}$$

□

Definition 4 (Indoor Trajectory Similarity Search). Given a set of indoor trajectories T, a query trajectory q, and an integer k, an *indoor trajectory similarity search* retrieves a set $S \subseteq T$ that consists of k trajectories, such that:

$$\forall x \in S \wedge \forall y \in T - S,\ SimTra(x, q) \geq SimTra(y, q)$$

□

Here, $SimTra(x, q)$ returns the similarity between x and q.

3 Indoor Trajectory Similarity

Given two indoor trajectories x and y, their similarity is defined as follows.

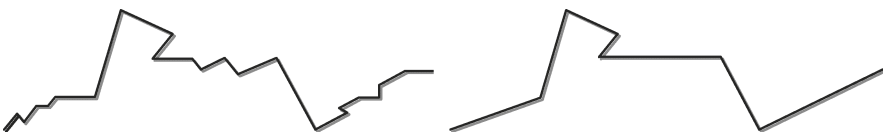
$$Sim(x, y) = a \cdot spaSim(x, y) + (1 - a) \cdot semSim(x, y) \quad (3.1)$$

Here, a is the weight reflecting the importance of the spatial similarity. Next, we give the details of measuring spatial similarity and semantic similarity.

3.1 Critical Point Based Spatial Similarity

We propose a critical-point-based solution for the measurement of spatial similarity for indoor trajectories. The idea of critical points can be illustrated by Fig. 1. As different points in a trajectory usually have different impacts in describing the most important features of the trajectory, e.g., direction, we can detect those important points, called *critical points*, and remove other non-critical points to make the original trajectory more simple and clear. As shown in Fig. 1(b), after constructing the critical-point-based trajectory for the original one in Fig. 1(a), the trajectory contains fewer points (original 23 points, only 8 critical points after transformation).

The benefits of critical points are two-fold. First, the computation time of similarity search can be reduced. Second, this design can be more effective than traditional methods like *Longest Common* Sub-sequence (*LCSS*) [15] and *Edit Distance* [9]. For example, the two trajectories shown in Fig. 1 will be regarded as dissimilar according to either the *LCSS* or *Edit Distance* approach. Note that the *LCSS* approach uses the longest common points to measure similarity (in this example, there are only 4 longest common points between the trajectories.), and the *Edit Distance* approach uses the count of edits transforming Fig. 1(a) to (b) to measure the similarity (in this example, fifteen edits have to be performed.). However, we can see from Fig. 1 that these two trajectories have very similar shapes and moving patterns (e.g., crossing through the hallway on a floor).



(a) The original trajectory (b) The critical-point-based trajectory

Fig. 1. The idea of critical points

One previous approach similar to the critical-point-based trajectory reduction is the Douglas-Peucker (DP) algorithm [25]. However, the DP algorithm is not effective in reducing complex indoor trajectories. For example, in Fig. 2, the trajectory from A to H will be reduced into A-E-H according to the DP algorithm, which omits some critical

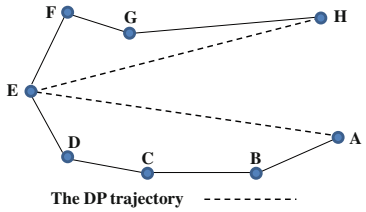


Fig. 2. The ineffectiveness of the DP algorithm in reducing indoor trajectories

points such as D and F. On the other hand, the trajectory of Fig. 2 is very common in indoor space, where each point can represent one room in a floor.

3.1.1 Critical Point

Definition 5 (Position Distance). Let p, p_c, and p_1 be indoor locations. The Euclidean distance from p to the line segment $\overrightarrow{p_c p_1}$ is called the *position distance* from p to p_c, denoted as $PD_{p \to p_c}$. □

Definition 6 (Position Angle). Let p_1, p_2, and p_3 be indoor locations. The angle between the line segment $\overrightarrow{p_1 p_2}$ and $\overrightarrow{p_2 p_3}$ is called the *position angle* of p_2, denoted as PA_{p_2} □

Figure 3 shows an example of *position distance (PD)* and *position angle (PA)*. Generally, we can find that a large position distance or a large position angle implies a substantial change on the moving direction (in Fig. 3 the referenced direction is $\overrightarrow{p_c p_1}$). Thus we can utilize the *position distance (PD)* and *position angle (PA)* to detect critical points in an indoor trajectory.

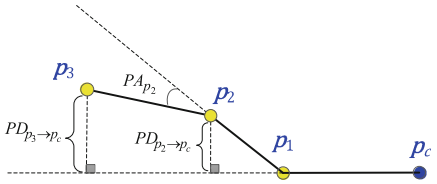


Fig. 3. An example of *position distance (PD)* and *position angle (PA)*

Definition 7 (Critical Point). Given an indoor trajectory x, an angle threshold θ and a distance threshold d, an indoor location p in x is called a *critical point* of x such that p satisfies one of the following criteria:

(1) p is the start or end point of x.
(2) p is an inter-connection location, e.g., an "*elevator*" room or a "*stair*" room, which can be identified through the labels in p.

(3) $PA_p > \theta$

(4) $PD_{p \to p_c} > d$, where p_c is the most-recently critical point preceding p in x.

□

The third condition is introduced to detect those points that lead to a substantial change on the movement direction. The last condition is used to find those points that gradually change the moving direction. We also identify inter-connection locations as critical points, e.g., elevator rooms and stair rooms, because those locations typically indicate dramatic changes on the moving direction. The initial critical point needed for computing *position distance (PD)* is the starting point of the trajectory.

Figure 4 shows the algorithm for detecting critical points of indoor trajectories.

Algorithm *CriticalPoint_Detection* (x, d, θ)

Input: $x=\{p_1, p_2, ..., p_n\}$ is an indoor trajectory, d is the distance threshold, and θ is the angle threshold.

Output: CP: the set of critical points.

```
1: CP ← {p1};
2: c ← p1 ; // c is the newest critical point, which is used for computing OD_{pi→c}
3: for i =2 to n−1 do
4:     if (pi is an interconnection location) or (OA_{pi} > θ) or (OD_{pi→c} > d) then
5:         CP←CP ∪ { pi };
6:         c ← pi ; // pi becomes the newest critical point for computing OD_{pi→c}
7:     end if
8:     CP ← CP ∪ {pn };
9: end for
10: return CP;
End CriticalPoint_Detection
```

Fig. 4. The algorithm to detect critical points

3.1.2 Indoor Distance

We now define the indoor distance. For each trajectory x, we can acquire its critical points using the algorithm in Fig. 4. Each two adjacent points form a line segment; then the trajectory can be represented as a sequence of three-dimensional line segments composed with these critical points.

Given two line segment L_i and L_j, in an indoor space (x, y, z), we suppose that L_j is on the (x, y) plane and L_i' denote the projection of L_i on the (x, y) plane. Then, we adapt the distance function defined in [18], which is proposed for two-dimensional Euclidean space, for three-dimensional indoor spaces. As illustrated in Fig. 5, the indoor distance between two line segments L_i and L_j is composed of four distances, namely the *perpendicular distance*, *horizontal distance*, *projection distance*, and *shifting distance*, which are defined as follows and based on the symbols shown in Fig. 5.

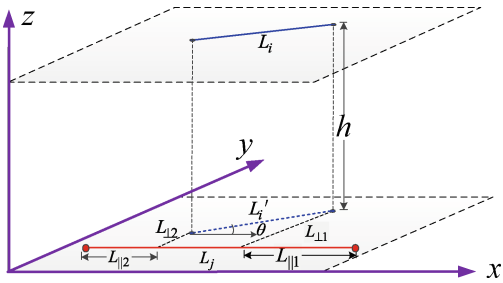


Fig. 5. Illustration of the indoor distance between two line segments

(1) The *perpendicular distance*: $d_\perp = \frac{L_{\perp 1}^2 + L_{\perp 2}^2}{L_{\perp 1} + L_{\perp 2}}$
(2) The *horizontal distance*: $d_\parallel = \min\left(L_{\parallel 1}, L_{\parallel 2}\right)$
(3) The *shifting distance*: $d_\theta = \mid L_j \mid \times \sin(\theta)$
(4) The *projection distance*: $d_h = h$

Then we can compute the distance between L_i and L_j by (3.2):

$$dist(L_i, L_j) = w_\perp \times d_\perp + w_{||} \times d_{||} + w_\theta \times d_\theta + w_h \times d_h \tag{3.2}$$

This distance function can accurately reflect the differences between the two segments, because they both consider the lengths and the directions of the segments. We can adjust the four weighting parameters, namely $w_\perp$,$w_\parallel$, w_θ, and w_h, according to real applications. In our experiment, we simply set these parameters to the same value.

Then, given two indoor trajectories x and y, whose number of locations are m and n, respectively, we can compute the indoor distance between x and y by (3.3).

$$distTR(x, y) = \begin{array}{c} dist(x.L_1, y.L_1) + min\{dist(x - \{x.L_1\}, y - \{y.L_1\}), \\ dist(x - \{x.L_1\}, y), dist(x, y - \{y.L_1\}) \end{array} \tag{3.3}$$

Finally, we define the spatial similarity between two trajectories x and y by (3.4). The $\frac{1}{1+distTR(x,y)}$ part is used to reflect the effect of indoor distance, where a farther distance means less similarity. The $\frac{\min(|x|,|y|)}{\max(|x|,|y|)}$ part is used to reflect the influence of the length of trajectories, which is based on the assumption that if the lengths of two trajectories are much different they are less similar.

$$spaSim(x, y) = \frac{1}{1 + distTR(x, y)} \times \frac{\min(|x|, |y|)}{\max(|x|, |y|)} \tag{3.4}$$

3.2 Hierarchical Semantic Similarity

We present the semantic classification tree, and then propose a new method to compute the hierarchical semantic similarity between trajectories.

3.2.1 Semantic Classification Tree

Generally, each indoor location has some descriptions, such as restaurant, movie theater, KTV, etc. These descriptions can have different granularities and levels. For example, the coarse-grained description by the label "*restaurant*" can be further explained using some fine-grained descriptions like "*Chinese restaurant*", "*Western restaurant*", etc. Therefore, we build a semantic classification tree to represent the hierarchical semantic relationship between the descriptions with different granularities. We denote such a tree as *SC_Tree*. Figure 6 shows an example of *SC_Tree*. In this tree, location classifications of different levels means different granularity of partition. Each non-leaf node in *SC_Tree* represents a category of locations, and each leaf node represents a semantic label of a location.

Fig. 6. An example of *SC_Tree*

Given a trajectory, we can find its corresponding leaf nodes in *SC_Tree*. For example, suppose that we have the following indoor trajectory (here each p_i represents a *sensorID* and a geographical coordinate, and for simplicity we omit the time information), the corresponding leaf nodes in *SC_Tree* in Fig. 6 are *p1*, *p2*, …, *p6*.

$$TRS = \langle (p_1, Nike), (p_2, Adidas), (p_3, LI-NING), (p_4, KFC), (p_5, Starbucks), (p_6, Lancome) \rangle$$

3.2.2 Hierarchical Semantic Patterns

Definition 8 (Hierarchical Semantic Pattern). Assume that *labelS* represents the labels of the leaf nodes in *SC_Tree*, *classS* is the set of labels of the non-leaf nodes, and H is the height of *SC_Tree,* the *hierarchical semantic pattern MP* of an indoor trajectory x is defined as $MP(x)$, which is a set of $MP_k(x), where\, 0 \le k \le H$. $MP_k(x)$ represents the semantic pattern of x at the k-th level in *SC_Tree*, which is defined as follows:

$$MP_k(x) = \begin{cases} \{\langle a_1, a_2, \ldots a_{|x|} \rangle | \forall i \in [1, |x|], a_i = x.p_i.label\}, & k = H \\ \{\langle a_1, a_2, \ldots a_{|MP_k(x)|} \rangle | \forall i \in [1, |MP_k(x)|], a_i \in (labelS \cup classS)\}, & k < H \end{cases}$$

□

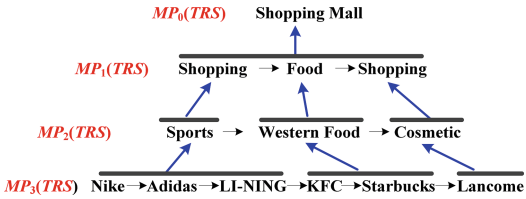


Fig. 7. An example for hierarchical moving patterns

For example, consider our trajectory *TRS*, its hierarchical semantic pattern is shown in Fig. 7. The lowest level of semantic pattern, i.e., $MP_3(TRS)$, is the sequence of semantic labels in *TRS*, while $MP_0(TRS), MP_1(TRS), MP_2(TRS)$ are sequences of non-leaf level labels.

3.2.3 Semantic Similarity

Given two indoor trajectory x and y, we first construct their hierarchical semantic patterns. Next, for the semantic pattern at each level of *SC_Tree*, we use the *LCSS* method [8] to calculate the similarity between the semantic patterns.

$$simMP(MP_i(x), MP_i(y)) = \frac{|LCSS(MP_i(x), MP_i(y))|}{min(|MP_i(x)|, |MP_i(y)|)} \tag{3.5}$$

Then, the semantic similarity of two trajectories T and R is defined.

$$semSim = \sum_{i=0}^{H} simMP(MP_i(x), MP_i(y)) \times w_i \tag{3.6}$$

Here, w_i is the importance of the similarity of ith semantic pattern to the total semantic similarity. This allows us to decrease the users' interests gradually from the leaf nodes up to the root, so that upper patterns contribute less to the users' interests. The following definition of w_i reflects this observation.

$$w_i = \frac{1}{\sum_{k=0}^{H} 2^k} \times 2^i \tag{3.7}$$

4 Experiment

We explain the experimental settings and then discuss the detailed results, including comparisons with several existing methods with respect to time performance and effectiveness.

4.1 Experimental Setup

Data Set. We simulate a shopping mall with two floors and generate indoor trajectory data using the indoor data generator *IndoorSTG* [20]. *IndoorSTG* can simulate different indoor spaces consisting of elements such as rooms, doors, corridors, stairs, elevators, and virtual positioning devices like RFID and Bluetooth readers. Besides, it can generate semantic-based trajectories for indoor moving objects. The simulated shopping mall has 94 indoor locations that represent different types of indoor elements such as rooms, elevators, corridors, and stairs. We simulate moving objects in such an indoor space during a time period of 20 days to generate different sets of trajectory data. The number of moving objects is set to 250, 500, 1,000, and 2,000, respectively, which yields 5,000, 10,000, 20,000, and 40,000 trajectories when the number of moving objects is varied from 250 to 2,000. We manually add the semantic label to each of the 94 indoor locations to represent the semantics. In the experiments, we use 20,000 trajectories as the default data set and randomly select 100 trajectories to serve as queries.

Metrics. We focus mainly on the time performance and effectiveness of similarity search. For time performance, we record the overall run time for processing queries when varying the number of trajectories. In order to evaluate the effectiveness, we use two metrics. The first is the *average distance* between the query trajectories and the results. The second is the *precision* of similarity search. Since we are using simulated data, it is not feasible to evaluate the precision from users' perspective. Instead, we utilize the idea of *Cumulative Gain* (*CG*) [21] to define precision. *CG* is well-known in information retrieval as part of the commonly-used metric *Normalized Discounted Cumulative Gain (NDCG)*, which is used to evaluate the relevance of search results.

Let q be a query trajectory and let R be the set of returned trajectories for q. We define the semantic relevance between q and a returned trajectory $r \in R$ as follows.

$$rel(r,q) = \sum_{i=1}^{\min(|r|,|q|)} \frac{1}{d(r_i, q_i)} \tag{4.1}$$

Here, $\frac{1}{d(r_i,q_i)}$ returns the relevance between two locations from q and r. And these two locations have the same sequence in q and r. When computing $rel(r,q)$, we first get the sum of the relevance between each location pair from q and r, and then map the sum into a value in the unit interval [0, 1]. We denote this normalized $rel(r,q)$ as $normalized\_rel(r,q)$. Considering that q and r may contain different number of locations, we use the shorter trajectory between q and r as the basic referential trajectory.

Given two locations r_1 and q_1, $d(r_1,q_1)$ returns the distance between the labels of the two locations in *SC_Tree*. As *SC_Tree* is constructed according to the semantics of locations, the distance between two nodes in *SC_Tree* is able to represent the semantic difference between the two locations.

If r_1 and q_1 have the same label then $d(r_1,q_1) = 1$. Otherwise, $d(r_1,q_1)$ is the number of nodes in the path from r_1 to q_1 in *SC_Tree* (including r_1 and q_1). For instance, in the *SC_Tree* shown in Fig. 6, $d(p_1,p_3) = 3$, while $d(p_1,p_4) = 7$.

As *SC_Tree* is organized in terms of the hierarchy of semantic classification, the closer two nodes are in *SC_Tree* the more similar their semantics are.

As a result, assume that q is a query trajectory and $R = \{r_1, r_2, \cdots, r_n\}$ is the set of returned trajectories for q, we give the computation of average distance and precision as follows:

$$average\_distance(q, R) = \frac{\sum_{0 \le i \le |R|} distTR(r_i, q)}{|R|} \tag{4.2}$$

$$precision(q, R) = \frac{\sum_{0 \le i \le |R|} normalized\_rel(r_i, q)}{|R|} \tag{4.3}$$

Here, $distTR(r_i, q)$ is defined in Formula (3.3), and $normalizde\_rel(r_i, q)$ is the normalized value of Formula (4.1) in the unit interval [0, 1]. In the experiments, we compute the mean average distances and precisions of 100 query trajectories as the final results.

Comparative Methods. We implement two classical algorithms as the baseline methods for measuring spatial similarity, *LCSS* (*Longest Common Subsequence*) [15] and *Edit Distance* [9]. They are denoted *LCSS_Indoor* and *ED_Indoor* in the following. Regarding semantic similarity, we use the *Cosine Similarity* [21] as the baseline method, which is one of the most popular methods of evaluating document similarity in information retrieval.

We call our method ***SIT*** (***S****imilarity of* ***I****ndoor* ***T****rajectories*), and we consider four variations of *SIT* in the experiments:

(1) *SIT_S* only considers spatial similarity and without introducing critical points.
(2) *SIT_SCP* only considers spatial similarity, but uses critical points.
(3) *SIT_SS* considers both spatial and hierarchical semantic similarities, but does not use critical points.
(4) *SIT_SSCP* considers both spatial and semantic similarities and uses critical points. For semantic similarity, it considers all the points in the trajectories.

4.2 Results

In the following, we present the results of the evaluation. For each experiment, we choose 100 trajectories as query trajectories from the dataset trajectories generated by *IndoorSTG*, and the results shown in the following are average results.

4.2.1 Precision

Figure 8 shows the precision on semantic similarity of all the methods. In this experiment, we set the parameter a to 0.5 and vary k from 5 to 30. Obviously, our methods *SIT_SS* and *SIT_SSCP* have much higher precision than *LCSS_indoor* and *ED_indoor*,

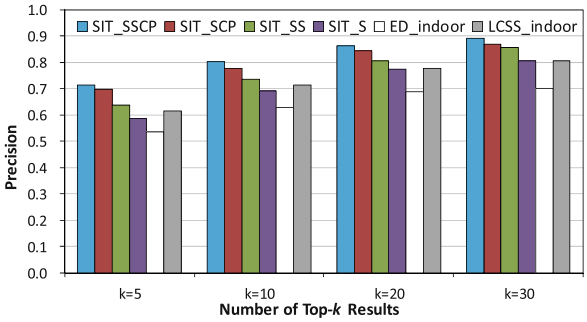


Fig. 8. Precision

which suggest that our proposal is effective. *SIT_S* and *SIT_SCP* perform poorly because they do not consider the influence of semantics.

In summary, *SIT_SSCP* and *SIT_SS* are effective for indoor trajectory similarity search as they consider both spatial and semantic similarities.

4.2.2 Average Distance

In this section, we evaluate the effectiveness of our proposal for spatial similarity search by calculating the average distance between the query trajectory and the trajectories returned.

As shown in Fig. 9, *SIT_S* has the smallest average distance, while *LCSS_indoor* and *ED_indoor* get the worst performance. This is mainly because *SIT_S* considers the features of indoor spaces and uses indoor distance. When considering critical points, *SIT_SCP* slightly larger the average distance compared with *SIT_S*. In fact, *SIT_SCP* can be regarded as an approximation of *SIT_S*, because it only considers the critical points of trajectories.

We can adjust the parameter a in Formula (3.1) to adapt special similarity-search needs of applications. For example, for user behavior analysis in shopping malls, semantic similarity may be important; thus, we can use a small value for parameter a. On the other hand, for public emergency monitoring in metro stations, spatial similarity could be more important, and we can use a high value for parameter a.

We can also see in Fig. 9 that *SIT_SSCP* and *SIT_SS* have higher average distances than *SIT_S* and *SIT_SCP*, which do not consider semantic similarity. Note that the returned set of top-k trajectories is influenced when we add semantic similarity into the computation of relevance. Therefore, the semantic similarity of the results will increase, but the spatial similarity (average distance) will decrease.

4.2.3 Time Performance

Figure 10 compares the running times of all the methods when executing 100 similarity searches on various number of trajectories. We vary the number of trajectories from 5,000 to 40,000, and calculate the time between issuing the queries and returning the ranked results.

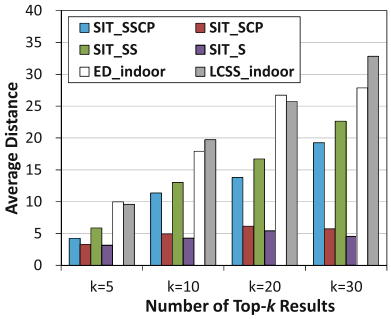


Fig. 9. Average distance

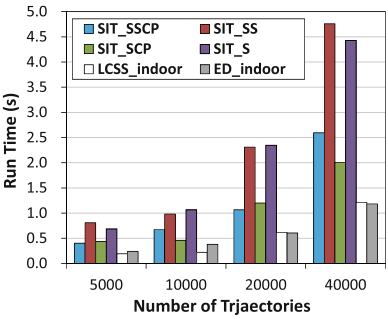


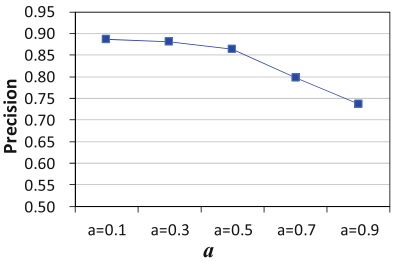
Fig. 10. Run Time

As Fig. 10 shows, the critical-point-based methods, including *SIT_SSCP* and *SIT_SCP*, performs faster than the methods that consider all points, namely *SIT_SS* and *SIT_S*. In particular, when increasing the number of trajectories, the benefit of critical points becomes more notable. On average, the critical-point-based methods are able to reduce the running times to about 50 % percent of run time compared with the methods without using critical points.

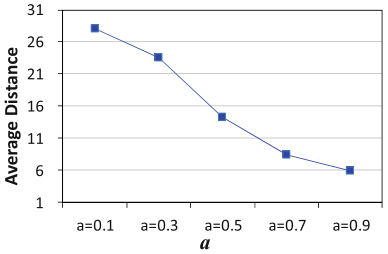
Both *LCSS_indoor* and *ED_indoor* get good time performance in the experiment, owing to their simple computation on distances and similarity measurement. However, as we have shown, they are not suitable for practical applications because of their poor effectiveness on both spatial similarity and semantic similarity.

4.2.4 Impact of Parameter a

Figure 11 shows the influence of parameter a on similarity measurement. Here, the fundamental algorithm is *SIT_SSCP*, and k is set to 20. This parameter is used to balance the impact of spatial and semantic similarity in the similarity evaluation. As shown in the figure, with the increase of a, the average spatial similarity decreases while the precision increases. This is simply because a large a means we give spatial similarity more weights in the computation of similarity. In real applications, we can tune this parameter to make it suit for the needs.



(a) Precision



(b) Average Distance

Fig. 11. Effect of parameter a

4.2.5 Comparison with Cosine Similarity

Cosine Similarity [21] is commonly used in information retrieval to evaluate document similarity. In this section, we aim to compare the performance of *Cosine Similarity* and our hierarchical semantic similarity. For this purpose, we modify *SIT_SSCP* by replacing the part of semantic similarity with *Cosine Similarity*. We denote this *Cosine Similarity* based method as *Cosine_SS*.

In order to computer *Cosine Similarity*, we perform the following procedure. First, each trajectory as well as the query trajectory is transformed into a vector representing the *Term Frequencies* (*TF*) of each semantic label in the trajectory. Next, we compute the *Cosine* value of the angle θ between vector A and vector B to measure the similarity between A and B. Here, A and B are vectors representing the term frequencies of the semantic labels in the trajectories.

For example, given the following two trajectories:

$$x = \langle (p_1, Nike), (p_2, LI-NING), (p_3, Adidas), (p_4, KFC) \rangle$$

$$y = \langle (p_1, Nike), (p_2, Adidas), (p_3, KFC), (p_4, Adidas), (p_5, Starbucks) \rangle$$

We first get the vectors of labels.

$$x' = \langle Nike, LI-NING, Adidas, KFC \rangle, \quad y' = \langle Nike, Adidas, KFC, Adidas, Starbucks \rangle$$

Then, we compute the term frequency for each semantic label, and get the vectors of term frequencies.

$$A = \langle 1, 1, 1, 1, 0 \rangle, \quad B = \langle 1, 0, 2, 1, 1 \rangle$$

Figures 12 and 13 compare precision and average distance, where parameter a is set to 0.5. Both figures show that *SIT_SSCP* performs better than *Cosine_SS*. Particularly, the precision of *SIT_SSCP* is about 1.1 times higher than that of *Cosine_SS*, and the average distance of *Cosine_SS* is over 4 times that of *SIT_SSCP*.

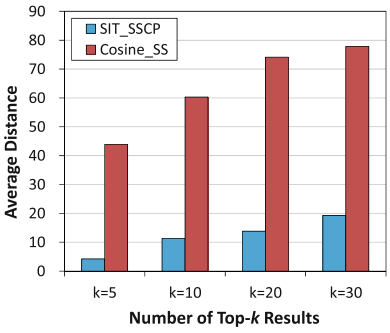


Fig. 12. Average distances of *SIT_SSCP* and *Cosine Similarity (Cosine_SS)*

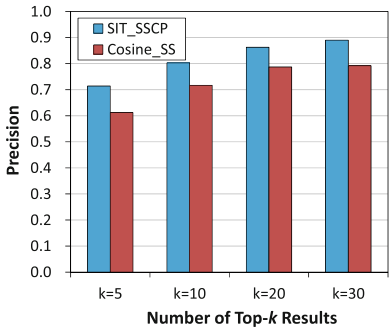


Fig. 13. Precisions of *SIT_SSCP* and *Cosine Similarity (Cosine_SS)*

5 Related Work

Previous efforts on moving-object trajectories mainly focus on outdoor space such as Euclidean space and road network space. Many approaches for measuring outdoor trajectory similarity have been proposed, including *Dynamic Time Warping* (*DTW*) [5], *One Way Distance* [6], *Longest Common Subsequence* (*LCSS*) [7, 8], *Edit Distance* based approaches [9, 10]. Although these approaches can also be used for indoor trajectory similarity search, they are not effective in indoor spaces because of the major difference between the computation of indoor distance and that of outdoor distance. Recently, some researchers begin to study the semantic similarity among trajectories [12–14].

However, trajectories in indoor spaces and outdoor spaces are different and most outdoor similarity measures have to be re-considered for indoor scenarios. Currently, there have been few studies for indoor trajectory similarity analysis. The only one that exactly focuses on indoor settings is called *CVTI* (*Common Visit Time Interval*) [15], which is actually based on the *LCSS* approach. As the *LCSS* approach has been proposed to analyze the similarity between strings or sequences, they make each character in strings corresponds to a cell id. It aims to find common time interval at the same location between two trajectories, and then use the common time intervals to define the similarity. If two trajectories stay at the same cell during the same time interval, they will be considered more similar than the case where there is no common time interval. However, this approach only concerns the common time intervals among trajectories, but neglects many other important factors such as closeness between trajectories as well as the semantics of trajectories.

Semantics of trajectories have attracted much attention in trajectory analysis [16–19]. Josh et al. [16] take into account the semantics of trajectories and propose a novel approach for recommending potential friends based on users' labels on trajectories in location-based social networks. They mine users' similarity from GPS trajectory data by considering semantic meanings of trajectories. Since the semantic labels of trajectories can reflect the preference and interests of users, many researchers propose to integrate semantics into trajectory analysis and further provide personalized location services or recommendations [17, 18]. For example, in [18], Haibo et al. propose to take users' preferences into consideration to provide personalized location searching.

In a trajectory, there are some points that can describe the spatial characteristics of the trajectory, such as turning points or others. These points are similar to the critical points proposed in this paper. Our proposal of critical points is inspired by [19], where they find that critical points are useful for region partitioning and location clustering. The major differences of our proposal and the work in [19] are two folds. First, we develop new algorithms suitable for indoor spaces to detect critical points in indoor moving trajectories. Second, we first use critical points in similarity search on indoor moving trajectories.

6 Conclusions

Similarity trajectory search is mostly based on Euclidean space or road network space before. They are not suitable for indoor spaces. In this paper, we present a new similarity measure considering both spatial and semantic similarities between indoor trajectories. We propose a critical-point-based method for spatial similarity as well as a hierarchical semantic pattern based method for semantic similarity. Comparative experiments suggest that our proposal is effective for indoor trajectory similarity search.

Our future work will focus on taking into account the time dimension [11] into indoor trajectory similarity search. Specifically, we will concentrate on users' stay times in indoor locations.

Acknowledgement. This work is supported by the National Science Foundation of China under the grant number 61379037.

References

1. Dudas, P., Ghafourian, M., Karimi, H.: ONALIN: ontology and algorithm for indoor routing. In: Proceedings of MDM, pp. 720–725 (2009)
2. Jin, P., Zhang, L., Zhao, J., Zhao, L., Yue, L.: Semantics and modeling of indoor moving objects. Int. J. Multimedia Ubiquit. Eng. **7**(2), 153–158 (2012)
3. Li, D., Lee, D.: A topology-based semantic location model for indoor applications. In: Proceedings of ACM GIS, pp. 1–10 (2008)
4. Jensen, C.S., Lu, H., Yang, B.: Graph model based indoor tracking. Mobile data management. In: Proceedings of MDM, pp. 17–24 (2008)
5. Berndt, D.J., Clifford, J.: Finding patterns in time series: a dynamic programming approach. In: Advances in Knowledge Discovery and Data Mining, pp. 229–248. AAAI/MIT Press (1996)
6. Lin, B., Su, J.: One way distance: for shape based similarity search of moving object trajectories. GeoInformatica **12**(2), 117–142 (2008)
7. Boreczky, J.S., Rowe, L.A.: Comparison of video shot boundary detection techniques. J. Electron. Imaging **5**(2), 122–128 (1996)
8. Vlachos, M., Kollios, G., Gunopulos, D.: Discovering similar multidimensional trajectories. In: Proceedings of ICDE, pp. 673–684 (2002)
9. Chen, L., Ozsu, M.T., Oria, V.: Robust and efficient similarity search for moving object trajectories. In: Proceedings of SIGMOD, pp. 491–502 (2005)
10. Wang, Y., Yu, G., Gu, Y., Yue, D., Zhang, T.: Efficient similarity query in RFID trajectory databases. In: Chen, L., Tang, C., Yang, J., Gao, Y. (eds.) WAIM 2010. LNCS, vol. 6184, pp. 620–631. Springer, Heidelberg (2010)
11. Yuan, Y., Raubal, M.: Measuring similarity of mobile phone user trajectories- a Spatio-temporal Edit Distance method. Int. J. Geogr. Inf. Sci. **28**(3), 496–520 (2014)
12. Pelekis, N., Kopanakis, I., Marketos, G.: Similarity search in trajectory databases. In: Proceedings of TIME, pp. 129–140 (2007)
13. Frentzos, E., Gratsias, K., Theodoridis, Y.: Index-based most similar trajectory search. In: Proceedings of ICDE, pp. 816–825 (2007)

14. Dodge, S., Weibel, R., Laube, P.: Exploring movement-similarity analysis of moving objects. SIGSPATIAL Special (SIGSPATIAL) **1**(3), 11–16 (2009)
15. Kang, H.-Y., Kim, J.-S., Li, K.-J., Hwang, J.-R.: Similarity measures for trajectory of moving objects in cellular space. In: Proceedings of ACM SAC, pp. 1325–1330 (2009)
16. Ying, J.J., Lu, E.H., Lee, W.-C., Weng, T.-C., Tseng, V.S.: Mining user similarity from semantic trajectories. In: Proc. of GIS-LBSN, pp. 19–26 (2010)
17. Ma, C., Lu, H., Shou, L., Chen, G.: KSQ: Top-k similarity query on uncertain trajectories. IEEE Trans. Knowl. Data Eng. **25**(9), 2049–2062 (2013)
18. Wang, H., Liu, K.: User oriented trajectory similarity search. In: Proceedings of UrbComp, pp. 103–110 (2012)
19. Lee, J.-G., Han, J., Whang, K.-Y.: Trajectory clustering: a partition-and-group framework. In: Proceedings of SIGMOD, pp. 593–604 (2007)
20. Huang, C., Jin, P., Wang, H., Wang, N., Wan, S., Yue, L.: IndoorSTG: a flexible tool to generate trajectory data for indoor moving objects. In: Proceedings of MDM, pp. 341–343 (2013)
21. Manning, C.D., Raghavanm, P., Schütze, H.: An Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)
22. Schafer, M., Knapp, C., Chakraborty, S.: Automatic generation of topological indoor maps for real-time map-based localization and tracking. In: Proceedings of International Conference on Indoor Positioning and Indoor Navigation (IPIN), pp. 1–8. IEEE CS (2011)
23. Zhang, D., Yang, L.T., Chen, M., Zhao, S., Guo, M., Zhang, Y.: Real-time locating systems using active RFID for internet of things. IEEE Syst. J. **PP**(99), 1–10 (2014)
24. Stojanović, D., Stojanović, N.: Indoor localization and tracking: methods, technologies and research challenges. Autom. Control Robot. **13**(1), 57–72 (2014)
25. Douglas, D., Peucker, T.: Algorithms for the reduction of the number of points required to represent a line or its caricature. Can. Cartographer **10**(2), 112–122 (1973)