Shape-based Similarity Query for Trajectory of Mobile Objects

Yutaka Yanagisawa Jun-ichi Akahani Tetsuji Satoh

NTT Communication Science Laboratories,
NTT Corporation

{yutaka,akahani,satoh}@cslab.kecl.ntt.co.jp
http://www.kecl.ntt.co.jp/scl/sirg/

Abstract. In this paper, we describe an efficient indexing method for a shape-based similarity search of the trajectory of dynamically changing locations of people and mobile objects. In order to manage trajectories in database systems, we define a data model of trajectories as directed lines in a space, and the similarity between trajectories is defined as the Euclidean distance between directed discrete lines. Our proposed similarity query can be used to find interested patterns embedded into the trajectories, for example, the trajectories of mobile cars in a city may include patterns for expecting traffic jams. Furthermore, we propose an efficient indexing method to retrieve similar trajectories for a query by combining a spatial indexing technique (R^+-Tree) and a dimension reduction technique, which is called PAA (Piecewise Approximate Aggregate). The indexing method can efficiently retrieve trajectories whose *shape* in a space is similar to the shape of a candidate trajectory from the database.

1 Introduction

Recently, many location sensors such as GPS have been developed, and we can obtain the trajectory of users and moving objects using these sensors [12]. Trajectory data are widely used in location-aware systems [1], car navigation systems, and other location-based information systems, that can provide services according to a user's current location. These applications have stored in them a lot of trajectories, and these trajectories may include interesting individual patterns of each user. For example, by analyzing trajectories of users who work in a building, we can find important passages, rooms, stairs, and other facilities that are used frequently. The result of the analysis can be used for the management and maintenance of the buildings. In the case of a navigation system, a driver can check the route to a city by referring to the trajectories of other users who have driven to the city before. In another case, we can study characteristics to improve performance in a sport by analyzing the motion data measured by the sensors attached to the bodies of top sports players.

There have been many studies on managing mobile objects data (MOD) [2] [8] [11] [14] [16]. One of the most interesting of these is the development of efficient method to retrieve objects, which is indicated by either a spatiotemporal *range query* or a spatio-temporal *nearest neighbor query*. Both queries are defined as the distance between the trajectory of a mobile object and an indicated point in a space. For example, the range query is generally defined as the query for retrieving all objects which passed within a given distance of an indicated point, such as "retrieve all of the people who walked within one mile of the buildings at the time." The range query can also be defined as the query to retrieve all objects that passed within an indicated polygon.

In both cases, the query is defined using the distance between figures in a space. These distance-based queries are useful in location management of mobile objects [15], however, these queries do not have enough power to analyze the pattern of the objects' motion. As mentioned above, because we are interested in the extraction of the individual moving patterns of each object from the trajectories, it is necessary to develop more powerful tools to analyze the trajectories. Hence, we propose *shape-based* queries of trajectories in space for the analysis, for instance, "retrieve all objects that have a similar shape to the trajectory where a user walked in a shop." Using this query, we may classify the customers in the shop based on their shape patterns of the trajectories. In other words, our approach is based on the shape similarity between lines, while the existing approaches adopt the distance between points as the key to retrieve required objects.

It is difficult to define the similarity between lines in a space. However, we found this useful idea through research of time series databases [6] [7] [10]. The time series database systems can store time series data such as temperature, economic indicators, population, wave signals, and so on, in addition to supporting queries for extracting patterns from the time series data. Most of the time series database systems adopt the Euclidean distance between two time data sequences [7] for analysis; if two sequences, c, c', are given as $\langle w_1, w_2, \ldots, w_n \rangle$ and $\langle w'_1, w'_2, \ldots, w'_n \rangle$, the similarity can be defined as

$$D(c, c') = \sqrt{(w_1 - w'_1)^2 + \ldots + (w_n - w'_n)^2}.$$

(In Section 3, we describe the similarity in detail) Because trajectory is a type of time series data, the time series database is able to deal with trajectory. However, trajectory not only has a time series data feature, but also has a directed line in space feature. For example, it is difficult for the time series database to find data for a geographic and spatial query.

Therefore, in this paper, we present a data model for trajectories of mobile data, and a query based on the distance between two trajectories by extending the similarity used in the time series database systems. Moreover, we propose a new indexing method for retrieving required trajectories by queries based on our defined distance between trajectories. In Section 2, we describe our proposed data model for the trajectory. Section 3 describes the distance between two discrete directed lines for calculating similarities between two trajectories . In Section 4,

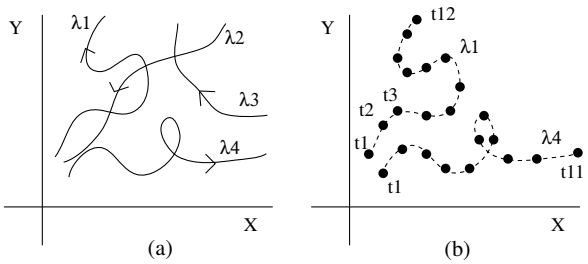

Fig. 1. Trajectory of Mobile Objects ((a) trajectory in the real world. (b) trajectory stored in a database)

we present both the processing method for our proposed query and an indexing technique that is an extension of Piecewise Aggregate Approximation (PAA) [7]. Finally the evaluation of our approach is shown in Section 5.

2 Trajectory of Mobile Objects

In order to effectively manage mobile objects, it is necessary to manage the location of each object at each time. Generally, a location management system can retrieve objects located in the indicated area at the indicated time [2]. However, we are interested in the similarity of the trajectory's shape in a space. In order to define the similarity between trajectories, it is necessary at first to define the trajectory as a figure drawn in space. Hence, we define the data model for the trajectory of mobile objects[1].

A real-world trajectory is a directed continuous line with a start and end point (Figure 1(a)). Given a two-dimensional space $\boldsymbol{R}^2$ and a closed time interval $I_\lambda = [t, t']$ with $t < t'$, a trajectory λ is defined as follows,

Definition 1 : *Trajectory*
A trajectory is the image of a continuous mapping of $\lambda : I_\lambda \rightarrow \boldsymbol{R}^2$.

This definition is a temporal extension of the definition of a simple line described in [3]. Next, we denote the length of trajectories in $\boldsymbol{R}^2$ as L_S and the interval of trajectories in temporal space as L_T:

Definition 2 : *Length of Trajectory in Space* $\boldsymbol{R}^2$
The length of trajectory λ during a period $[t_0, t_1]$ is denoted as $L_S(\lambda, [t_0, t_1])$ calculated as follows:

$$L_S(\lambda, [t_0, t_1]) = \int_{t_0}^{t_1} \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}\, dt, \text{ where } \lambda(t) = (x, y)$$

[1] For simplification of the problem, we just focus on the trajectory of mobile objects; in other words, we do not discuss the data model of the other attributes of the objects, such as shape, name, and so on.

The length of the whole trajectory is denoted as $L_S(\lambda)(= L_S(\lambda, [t, t']))$.

Definition 3 : *Temporal Interval of Trajectory*
The $\boldsymbol{x} = (x, y)$ is a vector in space $\boldsymbol{R}^2$. The temporal interval of trajectory λ between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ on λ is defined as follows:

$$L_T(\lambda, [\boldsymbol{x}_i, \boldsymbol{x}_j]) = |t_j - t_i|, \text{ where } \lambda(t_i) = \boldsymbol{x}_i, \lambda(t_j) = \boldsymbol{x}_j, \text{ and } t_i, t_j \in I_\lambda[t, t']$$

$$L_T(\lambda) = |t' - t|$$

However, a location sensor device such as GPS does not continuously measure the coordinates of a mobile object, but samples such data. The measured data are thus a sequence of coordinates of positions shown in Figure 1(b). Hence, we define discrete trajectory $\dot{\lambda}$ as a discrete function. Each vector $\boldsymbol{x}_i$ represents a position of a mobile object at each time $\boldsymbol{T}_{\dot{\lambda}} = \{t_0, t_1, \ldots, t_m\}$ in the space.

Definition 4 :*Discrete Trajectory*
A discrete trajectory is the image of a discrete mapping: $\dot{\lambda} : \boldsymbol{T}_{\dot{\lambda}} \rightarrow \boldsymbol{R}^2$.

A discrete trajectory can be represented as a vector sequence $\langle \boldsymbol{x}_{t1}, \ldots, \boldsymbol{x}_{tm} \rangle$, also. If $\boldsymbol{T}_{\dot{\lambda}} = \{1, 2, \ldots, m\}$, we denote the discrite trajectory $\dot{\lambda}$ as just a simple vector sequence $\langle \boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \rangle$. Additionally, where $\dot{\lambda}(t_i) = \boldsymbol{x}_i$, we introduce several notations; $\boldsymbol{T}_{\dot{\lambda}}(i) = t_i$, $\boldsymbol{X}_{\dot{\lambda}}(i) = \boldsymbol{x}_i$, and $|\dot{\lambda}|$ is the number of the vectors included in $\dot{\lambda}$ ($|\dot{\lambda}| = |\boldsymbol{T}_{\dot{\lambda}}|$). Next, we define the distance between two vectors $\boldsymbol{x}, \boldsymbol{x}'$ in $\boldsymbol{R}^2$.

Definition 5 :*Distance of Vectors*

$$D(\boldsymbol{x}, \boldsymbol{x}') = \sqrt{(x - x')^2 + (y - y')^2}$$

That is, this definition assumes that space $\boldsymbol{R}^2$ is Euclidean.

Although $\dot{\lambda}$ is a discrete line, it is necessary to deal with $\dot{\lambda}$ as a continuous line in a query. In order to satisfy this requirement, we define a function to convert the discrete line into a continuous line. There are various methods to calculate an approximate continuous line from a discrete line [4]. In our approach, we adopted the piecewise linear approximation because of its simplicity and popularity [15].

Definition 6 : *Piecewise Linear Approximation of* $\dot{\lambda}$
$\tilde{\lambda} : [t_0, t_m] \rightarrow \boldsymbol{R}^2$ is given as

$$\tilde{\lambda}(t) = \begin{cases} \dot{\lambda}(t) & \text{if } t \in \boldsymbol{T}_{\dot{\lambda}} \\ \frac{t - t_i}{t_{i+1} - t_i} \dot{\lambda}(t_i) + \frac{t_{i+1} - t}{t_{i+1} - t_i} \dot{\lambda}(t_{i+1}) & \text{if } t \notin \boldsymbol{T}_{\dot{\lambda}} \end{cases}$$

t_i must be selected under the condition: $t_i < t < t_{i+1}$

In the rest of this paper, we mainly discuss the features of both λ and $\dot{\lambda}$. Note that in this paper, we only mention the trajectory on $\boldsymbol{R}^2$, but our proposed model and techniques can obviously be adapted to the higher dimensional space $\boldsymbol{R}^n$.

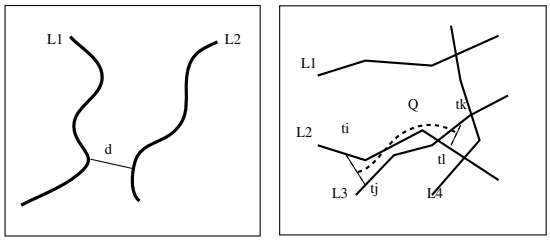

Fig. 2. Distance between trajectories

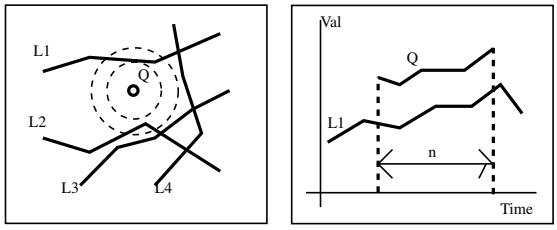

Fig. 3. Existing kNN approaches

3 Similarity Query based on Shapes of Lines

3.1 Shape-based Approach

The similarity query is useful in its own right as a tool for exploratory data analysis[7], and it is a significant element in many data mining applications. For instance, we may find the optimum arrangement of items in a market by analyzing the trajectories of customers walking around in a shop. In addition to its usefulness in the trajectory database, the similarity query is one of the most interesting fields in time series databases. In time series databases, the similarity between two sets of time series data is typically measured by the Euclidean distance [6] [7], which can be calculated efficiently.

However, there have been few discussions on the similarity between two lines in space because the previous approaches for spatial queries have focused on the "distance" between a point and a line [2] [9] [15]. The interest of the previous approaches was mainly to find objects that pass a point near the indicated point, such as a car passing through a street. On the other hand, we are interested in the "shape" of the trajectory. In order to calculate shape-based similarities among trajectories, it is necessary to define a new similarity for the trajectories, as shown in Figure 2(b).

In general, the similarity query is represented as a k Nearest Neighbor Query (kNN) [2] [5] [9] . There are two types of existing approaches, one is based on spatial similarities, the other is based on similarity between two time series data. The example of the existing spatial kNN is illustrated in Figure 3(a). In this case, the answer is L_1, L_2 when K is 2. On the other hand, the similarity between two time series data is defined as the Euclidean distance between two time series, where the length of each is n. The distance is defined as the Euclidean distance between two n-dimensonal vector data [7] shown in Figure 3(b). While this distance of the time series data is based on shape, the distance is defined only in the case of $\boldsymbol{R}^1 \times T$ ($T = [0, \infty]$), but not in the case of $\boldsymbol{R}^n \times T$, shown in Figure 2(b). Since the trajectory has both spatial and temporal features, we consider three types of similarity queries for trajectories as follows:

Spatio-Temporal Similarity: based on a spatio-temporal feature in $\boldsymbol{R}^2 \times T$.
Spatial Similarity: based on a spatial only feature in $\boldsymbol{R}^2$ without temporal features.
Temporal Similarity: based on a temporal only feature in $\boldsymbol{R}^1 \times T$ without spatial features.

In the rest of this section, we define the similarity in the first two cases. We do not define the temporal similarity because this similarity is the same as the similarity defined for the time series databases.

3.2 Shape-based similarity query

As mentioned above, the trajectory has a time series data feature. We define the similarity between two trajectories in the same manner as for the similarity defined in the time series query[7]. For the time series database, the similarity of the two time series data, where each has n value, is given by the Euclidean distance between vectors in $\boldsymbol{R}^n$. In [6] and [7], when there are two time series data, $c = \langle w_1, w_2, \ldots, w_n \rangle$, $c' = \langle w'_1, w'_2, \ldots, w'_n \rangle$, the distance $D(c, c')$ is defined as follows:

$$D(c, c') = \sqrt{(w_1 - w'_1)^2 + \ldots + (w_n - w'_n)^2}$$

This definition can be extended if each vector $\boldsymbol{x}$ is a vector in space $\boldsymbol{R}^2$, when the time series vectors are $\boldsymbol{X} = \langle \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \rangle$, $\boldsymbol{X'} = \langle \boldsymbol{x}'_1, \boldsymbol{x}'_2, \ldots, \boldsymbol{x}'_n \rangle$, and the distance is $D(c, c')$. We define the distance between two time series vectors $D(\boldsymbol{X}, \boldsymbol{X'})$ by extending the definition of $D(c, c')$, as follows:

$$D(\boldsymbol{X}, \boldsymbol{X'}) = \sqrt{D(\boldsymbol{x_1}, \boldsymbol{x'_1})^2 + \ldots + D(\boldsymbol{x_n}, \boldsymbol{x'_n})^2}$$

Based on this definition, we consider the shape-based similarity query for trajectories. Here, $\dot{\boldsymbol{\Lambda}}$ is the set of discrete trjectories stored in the database, and each $\dot{\lambda}_i$ ($\dot{\lambda}_i \in \dot{\boldsymbol{\Lambda}}$) is a discrete trajectory, such as $\dot{\lambda}_i = \langle \boldsymbol{x_1}, \ldots, \boldsymbol{x_m} \rangle$. The query trajectory $\dot{\lambda}_q$ is given as $\dot{\lambda}_q = \langle \boldsymbol{x_1}, \ldots, \boldsymbol{x_n} \rangle$. The shape-based range query can then be defined using $\dot{\boldsymbol{\Lambda}}$, λ_q, and the previous defined distance between two time series vectors, as follows:

Input : $\dot{\boldsymbol{\Lambda}}$, $\dot{\lambda}_q$ and θ (θ is a natural number).
Output : $\dot{\boldsymbol{\Lambda}}_a$, $\{\dot{\lambda}_{a1}, \ldots, \dot{\lambda}_{ak}\} \in \dot{\boldsymbol{\Lambda}}_a$.

function Q_{range}(θ: integer, $\dot{\lambda}_q$, $\dot{\boldsymbol{\Lambda}}$) : $\dot{\boldsymbol{\Lambda}}_a$
begin
  var j : integer, $l := |\dot{\lambda}_q|$, $\dot{\boldsymbol{\Lambda}}_a := \phi$;
  for each $\dot{\lambda}_i$ in $\dot{\boldsymbol{\Lambda}}$ **do**
    for $j := 1$ to $|\dot{\lambda}_i| - l + 1$ **do**
    begin
      $\dot{\lambda}_{ij}$ = subsequence($\dot{\lambda}_i$, j, l);
        { *This function will return a subsequence of the original sequence* $\dot{\lambda}_i$, *such as* $\langle \boldsymbol{x}_j, \boldsymbol{x}_{j+1}, \ldots, \boldsymbol{x}_{j+l-1} \rangle$, *each* $\boldsymbol{x} \in \dot{\lambda}_i$}
      if $D(\dot{\lambda}_q, \dot{\lambda}_{ij}) < \theta$ **then**
        Add $\dot{\lambda}_{ij}$ to $\dot{\boldsymbol{\Lambda}}_a$;
    end;
  return $\dot{\boldsymbol{\Lambda}}_a$;
end.

Fig. 4. The process of the shape-based range query of trajectories

Definition 7 : *Shape-based Range Query*
The process for calculation of the shape-based range query $Q_{range}(\theta, \dot{\lambda}_q, \dot{\boldsymbol{\Lambda}})$ is given in Figure 4. The range query is defined as a subsequence match of trajectories as shown in Figure 5.

In addition, the nearest neighbor query can be defined using our distance between trajectories. In our definition, the temporal features are not indicated in the query, however, we consider that the temporal features can be indicated independently from the range query. For example, a query "Q_{range}(θ, $\dot{\lambda}_q$, $\dot{\boldsymbol{\Lambda}}$) $\wedge$ 11:00 $< \boldsymbol{T}_{\dot{\lambda}_{ai}}(1) <$ 12:00" means retrieving subsequences $\dot{\lambda}_{ai}$ where the distance between $\dot{\lambda}_q$ and $\dot{\lambda}_{ai}$ is less than θ. Moreover, the first vector in $\dot{\lambda}_{ai}$ is measured within the interval [11:00, 12:00].

3.3 Spatio-temporal distance between two trajectories

Our defined distance $D(\boldsymbol{X}, \boldsymbol{X}')$ can be used only in the case where each vector $\boldsymbol{x} \in \boldsymbol{X}$ is measured by the same interval, that is $\Delta t = t_{i+1} - t_i$ ($i = 1, \ldots, n-1$), where t_i is an interval from the time when $\boldsymbol{x}_i$ is measured. However, each vector in the trajectory is not always measured by the same interval Δt because sensor devices often lose the data. For example, a discrete trajectory illustrated in Figure 6(a) has no measured vectors at $t = 5, 7, 9$.. Therefore, to calculate the similarity using our definition, we define a temporal normalized discrete trajectory $\dot{\lambda}_{\Delta t}$ for trajectory λ, as follows:

Definition 8 : *Temporal Normalized Discrete Trajectory*

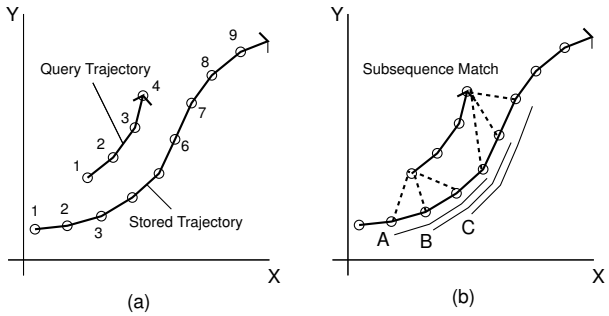

Fig. 5. Similarity query for trajectories

Given a trajectory λ defined for time interval $[t_S, t_E]$, and a natural number m, the temporal normalized discrete trajectory $\dot{\lambda}_{\Delta t}$ is defined as follows:

$$\dot{\lambda}_{\Delta t} = \langle \lambda(t_S), \lambda(t_S + \Delta t), \ldots, \lambda(t_S + m\Delta t) \rangle \text{ , where } t_S + m\Delta t = t_E$$

Intuitively, this discrete trajectory $\dot{\lambda}_{\Delta t}$ is the re-sampled trajectory per fixed interval Δt from λ, shown in Figure 6(c). In other words, $\dot{\lambda}_{\Delta t}$ is generated by dividing λ into equal interval Δt. For discrete trajectory $\dot{\lambda}$, we can use the piece-wise linear approximation $\tilde{\lambda}$ instead of λ. In the case of Figure 6, the temporal normalized discrete trajectory (Figure 6(c)) is generated from the approximate trajectory (Figure 6(b)).

Definition 9 : *Spatial-Temporal Similarity between two Trajectories*

Given two trajectories λ and λ' with the same temporal length (i.e. $L_T(\lambda) = L_T(\lambda')$) and a natural number m, the spatio-temporal distance (similarity) $D_{TS}(\lambda, \lambda')$ between λ and λ' is defined as follows:

$$D_{TS}(\lambda, \lambda') = \frac{1}{m+1} \sqrt{\sum_{i=0}^{m} D(\boldsymbol{X}_{\dot{\lambda}_{\Delta t}}(i), \boldsymbol{X}_{\dot{\lambda'}_{\Delta t}}(i))^2}, \text{ where } \Delta t = \frac{L_T(\lambda)}{m} = \frac{L_T(\lambda')}{m}$$

Note that $D_{TS}(\dot{\lambda}, \dot{\lambda}')$ can be defined as $D_{TS}(\tilde{\lambda}, \tilde{\lambda}')$. In this definition, the similarity is the Euclidean distance between trajectories represented as $m+1$ dimensional vectors, and the interval of each trajectory is normalized. Using this definition, it is possible to find trajectories whose shape is more similar to the query trajectory than can be found using previous methods.

3.4 Spatial distance between two trajectories

In definition 8, we focused on the shapes of the trajectories in space $\boldsymbol{R}^2 \times T$. However, there are cases where the shapes in $\boldsymbol{R}^2$ (without temporal features) are just as important, such as for example in the case of finding similar trajectories to those of a specified user when focusing on the "spatial" shape. Hence, we also define the spatial similarity between two trajectories.

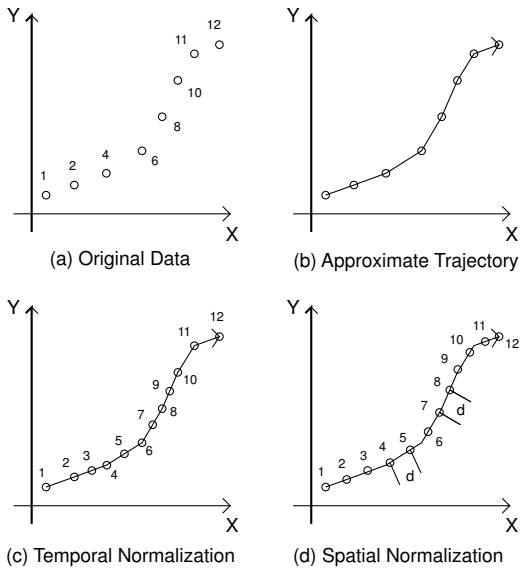

Fig. 6. Normalization of trajectories

Definition 10 : *Spatial Normalized Discrete Trajectory*

Given a trajectory λ and a natural number m, the spatial normalized discrete trajectory $\dot{\lambda}_\delta$ is defined as follows;

$$\dot{\lambda}_\delta = \langle \lambda(t_0), \ldots, \lambda(t_m) \rangle \text{ , where } L_S(\lambda, [t_{i-1}, t_i]) = \delta, (i = 1, \ldots, m)$$

Similar to $\dot{\lambda}_{\Delta t}$, $\dot{\lambda}_\delta$ is generated by dividing λ into equal spatial length δ. In the case of Figure 6, the spatial normalized discrete trajectory (Figure 6(d)) is generated from the approximate trajectory (Figure 6(b)).

Definition 11 : *Spatial Similarity between Trajectories*

Given two trajectories λ and λ' with the same spatial length (i.e. $L_S(\lambda) = L_S(\lambda')$) and a natural number m, the spatial distance (similarity) $D_S(\lambda, \lambda')$ between λ and λ' is defined as follows:

$$D_S(\lambda, \lambda') = \frac{1}{m+1} \sqrt{\sum_{i=0}^{m} D(\boldsymbol{X}_{\dot{\lambda}_\delta}(i), \boldsymbol{X}_{\dot{\lambda}'_\delta}(i))^2} \text{ where } \delta = \frac{L_S(\lambda)}{m} = \frac{L_S(\lambda')}{m}$$

Using this definition, it is possible to find the trajectories whose spatial shape is similar to that of the query trajectory without temporal features.

4 Indexing

With our proposed method for calculating similarity between trajectories, the database system can find the trajectories that have similar shapes to the shape

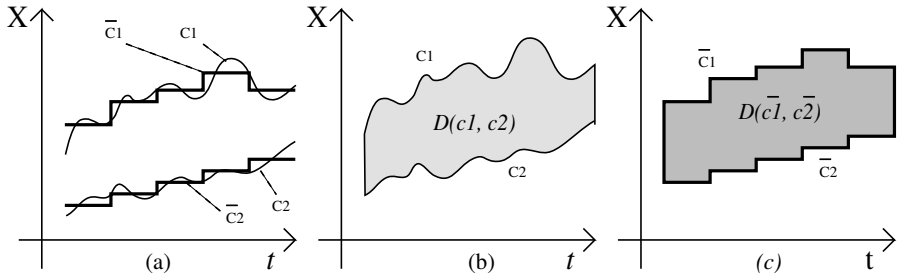

Fig. 7. Distance between two sequences c_1, c_2, and Distance between approximate sequences $\bar{c}_1$, $\bar{c}_2$

of the query trajectory. However, the cost of calculating our defined similarity is very high, because it is necessary to calculate Euclidean distances between each point on the query trajectory and each point on all trajectories stored in the database. In general, database systems store a lot of trajectories, and the amount of data is increasing rapidly. Therefore, it is important to reduce the cost of calculating similarities. In this section, we present an indexing method to reduce the cost of calculating similarities, which is based on techniques for reducing the dimensions of vector data.

4.1 Piecewise Aggregate Approximation

Piecewise Aggregate Approximation (PAA) [7] is a technique for reducing the cost of comparing two sets of time series data. The essential idea of this technique is the reduction of the number of compared data using the lower limit values of time series data. Here, we describe only an outline of this technique because it was fully presented in [7].

As mentioned in Section 3, the similarity between two time series data sets can be defined as Euclidean distance between two sequences represented as multi-dimensional vectors. Even if the query sequence has a shoter length m than the candidate sequence, the similarity can be defined as the distance between the query sequence and each subsequence of the candidate sequence, as illustrated in Figure 3(b). According to the definition of the similarity between two sequences (mentioned in Section 3.3), when the length of the query sequence is m and the maximum length of a candidate sequence is n, the order of calculating the similarity is obviously $O(mn)$ for each stored sequence. In order to reduce cost of comparison, it is necessary to reduce the number of compared values in sequence.

The PAA is a technique for generating approximate sequences to efficiently calculate similarity. If the original sequence has n values, the approximate sequence has only k values (k is a factor of n and k is much less than n). Each

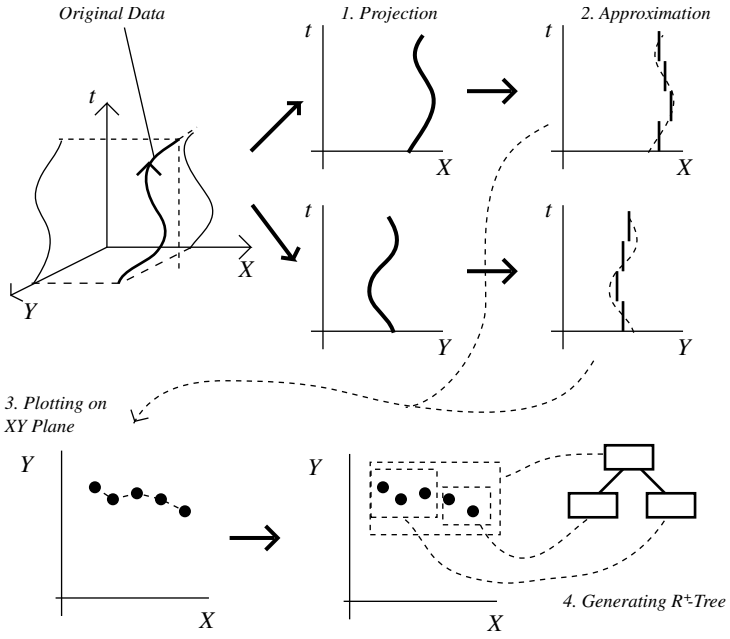

Fig. 8. The process for generating indexes to trajectories

member of the approximate sqeuence $\bar{c} = \langle \bar{w}, \ldots, \bar{w}_k \rangle$ is given as follows:

$$\bar{w}_i = \frac{1}{k} \sum_{j=k(i-1)+1}^{ki} w_j$$

In short, each $\bar{w}_i$ is calculated as the average of $\langle w_{k(i-1)+1}, \ldots, w_{ki} \rangle$. Moreover, it was proved in [7] that the approximate sequences $\bar{c}, \bar{c}'$ have a special relationship with the original sequences c, c' ($|c| = |c'| = k$):

$$D(\bar{c}, \bar{c}') < D(c, c')$$

This relationship means the distance between the approximate sequences is the lower limit of the distance between the original sequences. For example, in the case of Figure 7(a), the distance between c_1 and c_2 (Figure 7(b)) is always greater than the distance between approximate sequences $\bar{c}_1$ and $\bar{c}_2$, shown in Figure 7(c). Using this result, the database system can reduce the number of compared sequences with the query sequence.

4.2 Extended Indexing Method for Shape-based Similarity Query

The PAA is a simple and efficient technique for reducing the number of compared time series data; however, this technique has only been adapted to the data in

space $\boldsymbol{R}^1 \times T$. Hence, we extend this technique to trajectory data in space $\boldsymbol{R}^2 \times T$. Moreover, we present an efficient indexing method for trajectories by combining two techniques: PAA and a spatial indexing technique (R^+-Tree [13]) [2] We only describe the case of the spatial similarity query, but the essential idea can also be adapted to the spatio-temporal similarity query.

First, we define the extension of PAA for the 2-dimensional space:

Definition 12 : *2-Dimensional PAA (2D-PAA)*

Given a normalized spatial trajectory $\dot{\lambda}_\delta$ ($L_S(\dot{\lambda}_\delta) = n\delta$) and k, which is a factor of n, 2-Dimensional PAA of $\dot{\lambda}_\delta$ is:

$$\bar{\boldsymbol{X}} = \langle \bar{\boldsymbol{x}}_1, \bar{\boldsymbol{x}}_2, \ldots, \bar{\boldsymbol{x}}_k \rangle \text{ such that } \bar{\boldsymbol{x}}_i = (\bar{x}_i, \bar{y}_i)$$

$$\bar{x}_i = \frac{1}{k} \sum_{j=k(i-1)+1}^{ki} x_j, \ \bar{y}_i = \frac{1}{k} \sum_{j=k(i-1)+1}^{ki} y_j \text{ where } \boldsymbol{x}_j = (x_j, y_j)$$

Intuitively, 2D-PAA can approximate the trajectory's shape by calculating the center of the points contained in the trajectory. Using 2D-PAA and R^+-Tree, the indexes to trajectories can be generated as shown in Figure 8.

In order to retrieve trajectories whose distance to the query trajectory is less than θ, the database system processes the following steps;

1. Calculating $\bar{\boldsymbol{X}}_q = \langle \bar{\boldsymbol{x}}_{q1}, \ldots, \bar{\boldsymbol{x}}_{qk} \rangle$ for the query trajectory λ_q ($\lambda_{\delta q}$).
2. Searching sequences $\bar{\boldsymbol{X}}_1, \bar{\boldsymbol{X}}_2, \ldots$ (the length of each sequence is k) on R^+-Tree, such as $D_S(\bar{\boldsymbol{X}}_i, \bar{\boldsymbol{X}}_q) < \theta$.
3. Finding answer trajectories $\lambda_1, \lambda_2, \ldots$ such as $D_S(\lambda_i, \lambda_q) < \theta$.

In our approach, the number of compared data is reduced in two steps. The combination of both reduction techniques enables databases to retrieve answer trajectories efficiently. Therefore the trajectory databases can support the shape-based similarity query without a heavy load. Our indexing method's performance is evaluated in Section 5.

5 Performance Study

5.1 Experimental Settings

There are basically three variables that could affect the similarity between trajectories. The first is the length of a query, because as the length of query increases, the similarity decreases, but the cost of calculating similarities increases. The second variable is the density of the points in a space, since the similarity increases and the number of compared trajectories increases with the density of

[2] APCA [6], which is an extension of PAA, uses R-Tree based indexing techniques. However, the indexing method only uses R-Tree techniques to retrieve time series data (in space $\boldsymbol{R}^1 \times T$) efficiently, not to retrieve sequences of vectors such as trajectories in space $\boldsymbol{R}^2 \times T$.

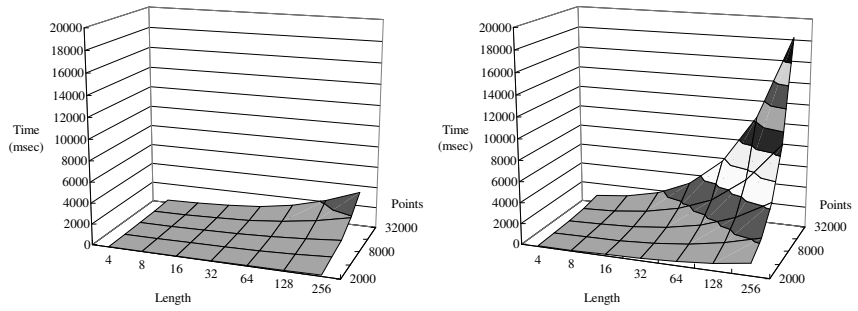

Fig. 9. The comparison of our proposed indexing method with the existing method

the points. The final variable is the complexity of the trajectory's shape. Generally, the irregularity of an object, motion causes greater complexity of shapes; in other words, people walking randomly generate the most complex shapes.

Therefore, we generate sample trajectories by changing these variables. The number of trajectories is 2-32, the length of trajectories is just 1000 (i.e., the maximum number of points is 32000), and the sampling interval Δt is fixed[3]. Each trajectory has shapes that represent people walking freely on a plane while changing speed and direction, and the frequency of the change in these values is altered in order to generate various complex shapes. In addition, the trajectories are embedded into a fixed area (size is 500×500) without any tendencies, in other words, the density of points can be controled by the number of embedded trajectories.

5.2 Efficiency

We have an experiment to assess the performance of our indexing method. For the experiment, we implemented two types of engine to find the trajectory which is the "nearest" to the query trajectory. The first engine checks every point on all trajectories (without any index), while the second engine checks only points filtered by our proposed index. We generated random trajectories for a query, and measured the calculation time required to find the nearest trajectory to the generated query trajectory. Since this is a simple performance evaluation of our approach, we gave a very simple query for each engine; the length of the query is fixed to k. In other word, the query trajectory $\dot{\lambda}_q$ has just k items, such as $\dot{\lambda}_q = \langle \boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \rangle$, and the approximate trajectory $\bar{\lambda}_q$ has only one item $\bar{\boldsymbol{x}} = (1/k \sum_{i=1}^{k} x_i, 1/k \sum_{i=1}^{k} y_i)$ where $\boldsymbol{x}_i = (x_i, y_i)$.

Figure 9 illustrates the experimental result in the case where the shape of trajectories is simple. While the left graph in the figure shows the calculation

[3] Even in the case where Δt is not fixed, we can take similar results from the case where Δt is fixed, because fixed trajectories can be mechanically generated from original trajectories with the piecewise linear approximation defined in Section 2.

time with our indexing methods, the right graph shows without any index. With our proposed index, the calculation time is 60%–75% less than without the index in each situation. For example, in the case where the query length is 256 and the number of points is 32000 (it is the worst case), the calculation time is 3,325 msec with the index. On the other hand the calculation time without the index is 18,206 msec. Although the time taken to generate indexes (overhead) is 3,377 msec when the number of points is 32,000, it is less than the time required without the index. As a result of our experiment, the advantage of our indexing method is clear.

In addition, we measured the calculation time in cases where the shape of trajectories is very complex, such as trajectories where each object moves almost randomly. In this case, the calculation time is 15%–20% greater than the time in the simple case; however, the rate of increase is the same in cases with the index and without the index.

6 Conclusion

The main contribution of this paper is the presentation of an efficient indexing method for processing shape-based similarity queries for trajectory databases. In order to calculate similarity between trajectories, we defined discrete trajectories that were re-sampled per fixed interval. Furthermore, we described the performance of our proposed indexing method, and we show the advantage of our method over the existing methods.

As future work, we will implement the trajectory database system to evaluate our proposed model, Furthermore, we will develop several real application programs such as a car navigation system, a personal navigation system, and other location-wares.

References

1. G. Chen and D. Kotz. Categorizing binary topological relations between regions, lines, and points in geographic databases. Technical Report TR2000-381, A Survey of Context-Aware Mobile Computing Research, Dept. of Computer Science, Dartmouth College, 2000.
2. H. Chon, D. Agrawal, and A. E. Abbadi. Query processing for moving objects with space-time grid storage model. In *MDM2002 Conference Proceedings*, pages 121–129, 2002.
3. E. Clementini and P. D. Felice. Topological invariants for lines. *IEEE Transaction on Knowledge and Data Engineering*, 10(1):38–54, 1998.
4. L. E. Elsgolc. *Calculus of Variations*. Pergamon Press LTD, 1961.
5. E. G. Hoel and H. Samet. Efficient processing of spatial queries in line segment databases. In O. Gunther and H. J. Schek, editors, *SSD'91 Proceedings*, volume 525, pages 237–256. Springer-Verlag, 1991.
6. E. Keogh, K. Chakrabarti, S. Mehrotra, and M. Pazzan. Locally adaptive dimensionality reduction for indexing large time series databases. In *SIGMOD2001 Conference Proceedings*, pages 151–162, 2001.

7. E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286, 2001.
8. G. Kollios, D. Gunopulos, and V. J. Tsotras. On indexing mobile objects. In *SIGMOD'99 Conference Proceedings*, pages 261–272, 1999.
9. G. Kollios, V. J. Tsotras, D. Gunopulos, A. Delis, and M. Hadjieleftheriou. Indexing animated objects using spatiotemporal access methods. *IEEE Transactions on Knowledge and Data Engineering*, 13(5):758–777, 2001.
10. Y.-S. Moon, K.-Y. Whang, and W.-S. Han. General match: A subsequence matching method in time-series databases based on generalized windows. In *SIGMOD 2002 Conference Proceedings*, pages 382–393, 2002.
11. K. Porkaew, I. Lazaridis, and S. Mehrotra. Querying mobile objects in spatiotemporal databases. In C. S. Jensen, M. Schneider, B. Seeger, and V. J. Tsotras, editors, *SSTD 2001*, volume 2121 of *Lecture Notes in Computer Science*, pages 59–78. Springer-Verlag, 2001.
12. N. Priyantha, A. Miu, H. Balakrishnan, and S. Teller. The cricket compass for context-aware mobile applications. In *MOBICOM2001 Conference Proceedings*, pages 1–14, 2001.
13. T. Sellis, N. Roussopoulos, and C. Faloutsos. The R^+-tree: A dynamic index for multidimensional objects. In *VLDB'87 Conference Proceedings*, pages 3–11, 1987.
14. A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and querying moving objects. In *ICDE'97 Proceedings*, pages 422–432, 1997.
15. M. Vazirgiannis and O. Wolfson. A spatiotemporal model and language for moving objects on road networks. In C. S. Jensen, M. Schneider, B. Seeger, and V. J. Tsotras, editors, *SSTD 2001*, volume 2121 of *Lecture Notes in Computer Science*, pages 20–35. Springer-Verlag, 2001.
16. O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang. Moving objects databases: Issues and solutions. In *Statistical and Scientific Database Management (SSDM'98) Conference Proceedings*, pages 111–122, 1998.