Predictive Indexing for Position Data of Moving Objects in the Real World

Yutaka Yanagisawa

NTT Communication Science Laboratories

Abstract. We propose a spatial-temporal indexing method for moving objects based on a prediction technique using motion patterns extracted from practical data, such as trajectories of pedestrians. To build an efficient index structure, we conducted an experiment to analyze practical moving objects, such as people walking in a hall. As a result, we found that any moving objects can be classified into just three types of motion characteristics: 1) staying, 2) straight-moving, 3) random walking. Indexing systems can predict highly accurate future positions of each object based on our found characteristics; moreover, the indexing system can build efficient MBRs in the spatial-temporal data structure. To show the advantage of our prediction method over previous works, we conducted an experiment to evaluate the performance of each prediction method.

1 Introduction

In recent years, we have been able to use highly accurate positioning devices to track moving objects, such as pedestrians and cars. The position is one of the most significant data for extracting contexts from the real world. Then many context-aware services use the position data for providing services [1], [2]. The Moving Object Database (MoDB) [3] is a database system that can manage position data of real moving objects. Cost reductions in managing such trajectories are one of the most significant challenges for applications using position data. Various types of efficient data structures have been proposed [4] [5] [6] for managing trajectories.

In general, a position is denoted as $p = \{o, t, x, y\}$, which means object o is located at point $\langle x, y\rangle$ at time t, and trajectory λ of a moving object is also denoted as a sequence of positions $\langle p_0, \ldots, p_n\rangle$. Obviously, trajectory can be represented as a model of spatial and temporal data. Thus, most previous MoDBs adapt traditional tree-based indexing mechanisms, such as R-tree [7], which uses Minimum Bounding Rectangles (MBRs) for managing trajectories and the positions of each moving object. However, such an MoDB must frequently update both the tree structure and MBRs because every object continuously changes its position second by second. The increase in the update cost is one of the most serious problems with MoDBs.

To solve this problem, several MoDBs adapt a *predictive* indexing mechanism [8] [9] that calculates *predicted* MBRs, including the predicted future positions of

moving objects. The introduction of the mechanism enables MoDBs to manage positions without frequently updating MBRs. To greatly reduce the update cost, the mechanisms must predict the future positions of objects as accurately as possible. In this paper, therefore, we propose a new technique to accurately predict the future position of moving objects, and we also describe an improved indexing mechanism to manage the positions based on predicted MBRs.

To improve prediction accuracy, we investigated the features of the real trajectories obtained in our experiments. From the investigation results, two special motion patterns are found from trajectories: "staying and "straight-moving." Staying means that an object almost comes to a stop at a point for a period; on the other hand, straight-moving means an object moves in a straight line. Thus, we present a prediction technique based on these two motion patterns and "random-moving," which can represent any motion of objects.

Section 2 describes the trajectories obtained in our experiments. In Section 3, we explain both the found motion pattern and prediction function to calculate the future positions of moving objects. Moreover, in Section 4, we cite the performance of our proposed prediction technique by comparison with previous existing index structures. Finally, Section 5 concludes our work.

2 Trajectory Data

2.1 Moving Objects in Real World

For improving prediction techniques, we experimentally analyzed the trajectories of various types of moving objects, such as cars, people, and parts of human bodies. In this section, we focus on the following characteristics of three types of moving objects.

Trajectories of Moving Vehicles (Vehicle Data) For such data, we obtained the trajectories of working rickshaws in Nara, a famous former capital of Japan. We placed a GPS receiver on each rickshaw that recorded the points where the rickshaw was located every second. The errors of the GPS receivers are within 5 m. The average trajectory length is about 18 km, and the average trajectory period is about nine hours. Example rickshaw trajectories are illustrated in Figure 1. The lines in the figure show the trajectories of rickshaws moving in the northern part of Nara for nine hours a day.

The shape of this trajectory includes many *long straight lines* because a rickshaw moves along streets whose shape is almost a straight line. On the other hand, since a rickshaw waits at intersections for passengers, it tends to stay long at one place. Generally, the trajectories of such moving vehicles as taxis, buses, and trucks have the same characteristics as the trajectories of rickshaws.

Trajectories of Wandering Visitors (Visitor Data) We did an experiment to obtain the trajectories of 200 visitors to an exhibition that had about 100 booths. The size of the exhibition hall was 20 m $\times$ 40 m. We set 10 Lazar sensors

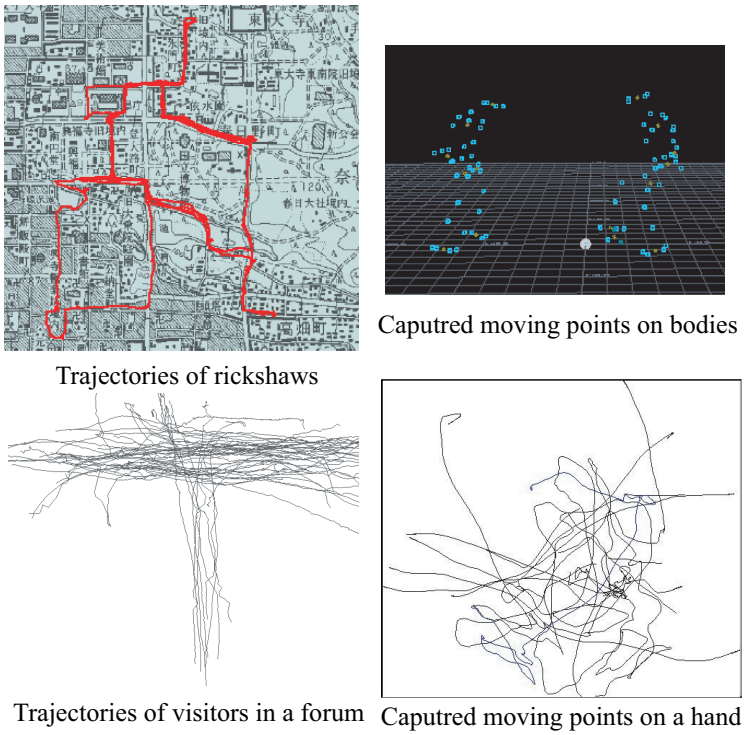

Fig. 1. Captured Moving Points

and five video cameras for tracking visitors in the hall, and each sensor could obtain the locations of visitors every 1/30 of a second. The average duration of all trajectories was about 60 minutes, and the average geographical length of the trajectories was about 50 m. The maximum error of a Lazar sensor is less than 1 m.

Because Lazar sensors lose visitors hidden by other visitors, we completed the incomplete trajectories with video data captured by hand. Figure 1 also illustrates the trajectories of visitors who walked in the hall during a five-minute period.

Three types of characteristic shapes were found in this type of trajectory: a gentle curved line, a short line, and so on. Because a visitor often walks and stops at booths, trajectory shapes tend to include such characteristic shapes. We also found another characteristic: the velocities of visitors differ, since various types of visitors meander looking for interesting items in the hall. The trajectories of visitors in an exhibition hall are similar to the visitors in museums, large shopping malls, art galleries, and so on.

Trajectories of Body Parts in Sports (Sport Data) We obtained the trajectory data of track points on the bodies of two soccer players using an optical

motion capture system. Each player had 36 track points on his/her body and the soccer ball had two track points, and the motion capture system tracked 74 points every 1/120 of a second. The time of all trajectories was two minutes, and the average geographical length of trajectories was about 2 m. The top right image in Figure 1 shows example trajectories of the left legs of the two soccer players when fighting for the ball. This figure shows a projection of trajectories from 3D space to a 2D plane, but the features of the data are the same in each dimension. This figure has eight trajectories because we obtained four sets of trajectories from two players. Each player moves in an area 6×6 m^2. These trajectories have many curves and turns but only a few straight lines. The velocities of these moving points are not fixed, and each point can suddenly accelerate or decelerate.

2.2 Motion Patterns

We found several characteristic motion patterns of practical moving objects in the trajectories we obtained. The motion patterns suggest that "when an object moves in a particular manner, we can predict its future motion." In our experiments, we found two basic patterns: staying and straight-moving.

Staying When an object almost stops at a place for a period, we describe it as *staying*. We did not find staying objects in the sport data, but 2/5 objects in the vehicle data were staying, and 9/10 objects of the visitor data were staying.

Moving Straight: When an object is moving in a straight line and its velocity is almost fixed, we say the object is *moving straight*. We found this motion pattern in all types of moving objects. Especially since 3/10 objects in the vehicle data are moving straight, the ratio is higher than in other data.

We can classify 7/10 – 9/10 of the objects in any data into these two patterns; however, the rest of the objects cannot be classified into any patterns. To classify all objects, we define one more motion pattern called "Moving Randomly" as follows.

Moving Randomly: When an object is continuously moving in unfixed directions at unfixed velocities, we say the object is *moving randomly*. In practical data, most such objects move in unfixed directions at almost fixed velocities. Such objects are found in visitor and sport data, but rarely in vehicle data.

We can classify any object based on our three defined patterns; moreover, the future motion of any object can be predicted by the definition of motion patterns.

2.3 Noise in Moving

The trajectories of moving objects often have low frequency noise because of positioning errors. Generally, existing databases deal with noiseless data but it

is difficult to clean up practical noisy trajectory data. To apply database systems to practical data, in this paper we describe the prediction of the future position of a moving object with such low frequency noise. Trajectory noise has two principal sources: positioning devices and the size of moving objects. Because no positioning device can specify an object's position without errors, trajectories inevitably have errors. The size of the object, moreover, causes noise because devices generally cannot decide where an object's center point is on its surface. For example, errors of laser sensors when tracking walking people are less than 50 cm because the sensor rarely decides the person's center point, and the horizontal area of a person is a circle whose radius is less than 50 cm.

We define maximum noise as the sum of these two errors; for instance, if positioning error is 1 m and error size is 50 cm, maximum error is calculated as 1.5 m. Maximum error is denoted as θ_p, which is an actual measurement.

3 Motion Prediction

In this section, we describe functions that predict an object's future point and how to apply prediction techniques to practical moving objects. Here, we consider our indexing technique is applied to both the nearest-neighbor query and the spatial-temporal range query.

3.1 Formalization of Motion Patterns

Before describing functions, we define motion patterns using mathematical equations. We denote a trajectory that includes the points of a moving object from time $t-m$ to t as $\lambda_{t,-m}$. If trajectory $\lambda_{t,-m}$ satisfies condition C, the trajectory's moving object has motion pattern C from time $t-m$ to t. We define three conditions, C_{st}, C_{sw}, and C_{rw}, for each motion pattern mentioned in Section 2.

Staying (C_{st}) We denote a position vector in $\lambda_{t,-m}$ at time i as $\boldsymbol{p}(i)$ for defining 'staying' condition C_{st} as $|\boldsymbol{p}(i) - \boldsymbol{p}(t)| = 0$ where $i = t-m, t-m+1, \ldots, t-1$, t. Condition C_{st} means that the maximum velocity of a moving object equals 0 from time $t-m$ to t.

Actually, practical data have noise θ_p, as mentioned in the previous section, so no practical objects completely stop at a place in the data. We introduce the influence of θ_p to the condition with the next extended equation:

$$\begin{aligned} C_{st} &: |\boldsymbol{p}(i) - \boldsymbol{p}(t)| < \theta_p \\ \boldsymbol{p}(i) &\in \lambda_{t,-m}. \end{aligned} \tag{1}$$

Here $|\boldsymbol{p}|$ means the vector length of $\boldsymbol{p}$. If an object moves less than distance θ_p from point $\boldsymbol{p}(t)$ during period $t-m$ to t, the object has satisfied the "staying" condition.

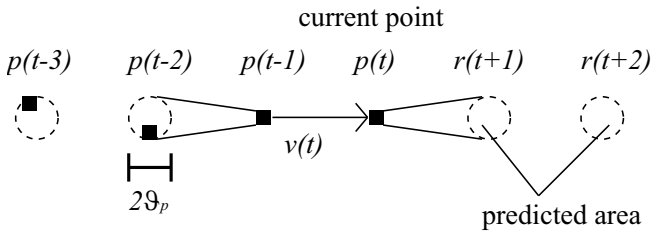

Fig. 2. Prediction for straight moving

Moving Straight(C_{sw}): We denote a velocity vector of an object in $\lambda_{t,-m}$ at time $i(0 \leq i \leq m)$ as $\boldsymbol{v}(i) = \boldsymbol{p}(i) - \boldsymbol{p}(i-1)$ for defining the 'moving straight' condition C_{sw} as $\boldsymbol{v}(i) = \boldsymbol{v}(t)$. This condition means that the difference between every velocity vector in $\lambda_{t,-m}$ and velocity vector $\boldsymbol{v}(t) = \boldsymbol{p}(t) - \boldsymbol{p}(t-1)$ at t always equals 0. Similar to the staying condition, we also considered the influence of noise θ_p. The actual conditions can be defined by:

$$\begin{aligned} &C_{sw} \ : \ |\boldsymbol{v}(i) - \boldsymbol{v}(t)| < \theta_p \\ &\boldsymbol{v}(i) = \boldsymbol{p}(i) - \boldsymbol{p}(i-1). \end{aligned} \tag{2}$$

Moving Randomly(C_{rw}) We classify an object to this pattern when it does not satisfy the previous two conditions: C_{st} and C_{sw}. But objects 'moving randomly' do not move freely on a plane since physical restrictions limit their maximum velocity; for example, no person can walk at 50 m/s. In our method, we must define the maximum velocity of objects as v_{max} for the 'moving randomly' condition.

Obviously, an object moves within a circle such that its center is $\boldsymbol{p}(t)$ and its radius is iv_{max}, where maximum velocity is v_{max} and the end point of the object at t is fixed to $\boldsymbol{p}(t)$. Therefore, ideal 'moving randomly' condition C_{rw} can be defined as $C_{rw} : |\boldsymbol{v}(i)| \leq iv_{max}(t-m \leq i \leq t)$, where maximum velocity is v_{max} in trajectory $\lambda_{t,-m}$. Condition C_{rw}, including the influence of θ_p, is defined by:

$$C_{rw} : |\boldsymbol{v}(i)| \leq v_{max} + \theta_p. \tag{3}$$

3.2 Predictive Function

To predict the future position of moving objects, we define functions for each condition.

Notation $R(j)$ means an area where an object of $\lambda_{t,+n}$ will move from time current t to future time $j = t + n$. If an object in trajectory $\lambda_{t,-m}$ satisfies condition C, we can calculate $R(j)$ for the object using the equations in C.

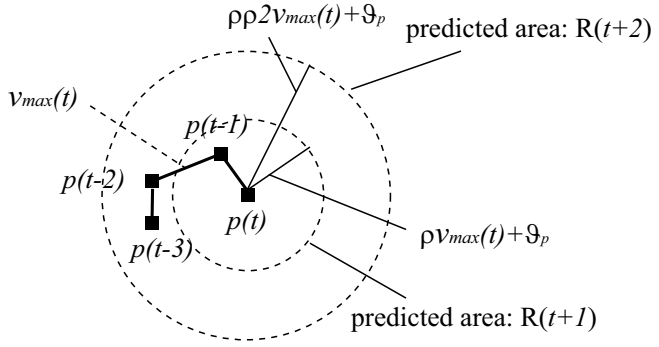

Fig. 3. Prediction for random moving

$R(j)$ is a closed area, and the shape of $R(j)$ is either a rectangle, a circle, or a combination of such diagrams. Hence, the calculation of $R(j)$ can be represented as function f using $\lambda_{t,-m}$ and time j, for example, $f(\lambda_{t,-m}, j) = R(j)$. We call f a "predictive function."

Note that previous work defined functions that give a future point of a moving object; however, it is actually impossible to determine a future place of an object in just a point. Therefore, we define f as a function that gives an area.

- Staying
  Because $C_{st} : |\boldsymbol{p}(i) - \boldsymbol{p}(t)| < \theta_p$, we define function f_{st} for staying as $R_{st}(j) = f_{st}(\lambda_{t,-m}, j)$, which is a circle such that its center point is $\boldsymbol{p}(t)$ and the radius is θ_p. The predicted area's actual shape is a rectangle because rectangles are available to build indexes based on MBRs.
- Moving Straight
  The function for moving straight is denoted as $R_{sw}(j) = f_{sw}(\lambda_{t,-m}, j)$. Figure 2 shows an example prediction area such that its shape is a circle whose center point is $\boldsymbol{p}(t) + (j - t)\boldsymbol{v}(t)$ and whose radius is θ_p. Similar to staying, we use a rectangle as the actual shape prediction area.
- Moving Randomly
  Based on the definition of condition C_{rw}, prediction area $R_{rw}(j) = f_{rw}(t + n)$ is a circle whose center point is $\boldsymbol{p}(t)$ and whose radius is $nv_{max} + \theta_p$, as illustrated in Figure 3. But this function often gives redundant area, especially because maximum velocity greatly increases the area. To avoid this problem, we reduce the outside of the area where the object seldom reaches.
  When an object moves randomly at a velocity less than v_{max} during period n seconds, accuracy ρ in which the object exists in a circle whose center point is $\boldsymbol{p}$ and whose radius is $r(0 \leq r \leq nv_{max})$ is given by the following equation:

$$\rho = \left(\frac{r}{nv_{max}}\right)^n. \tag{4}$$

To predict the area at $t+n$ within accuracy ρ, we decide the radius of the predicted circle as $nv_{max}\sqrt[n]{\rho}+\theta_p$. For example, where $\rho = 0.7$ and $n = 5$, the radius is decided as $0.93 \times 5v_{max}$. In practical trajectory data, v_{max} is larger than the effective velocity, so we use accuracy $\rho = 0.7$ in our evaluation, as mentioned in Section 4.

3.3 Prediction

For calculating the future area of an object at time $t+n$, the prediction system examines trajectory $\lambda_{t,-m}$, whose C_{st}, C_{sw}, or C_{rw} conditions are satisfied by the trajectory. Next, the system applies a function that corresponds to the condition. Because an object can be classified into either condition as mentioned, we can calculate its predicted area. If an object can be classified into both conditions C_{st} and C_{sw}, then the system applies condition C_{st} since the size of area $R_{st}(j)$ is less than $R_{sw}(j)$.

3.4 Construction of MBRs

To manage trajectory data, we adapt a traditional spatial data structure based on R-Tree in multidimensional space, which is similar to TPR- and STP-Trees. In traditional databases, because a set of data is added into a database continuously and randomly, the database must reconstruct the data structure every time a data set is added. Generally, such databases cannot construct the optimal data structure. But moving object data are periodically and simultaneously added to a database because positioning devices periodically obtain an object's position. We consider a moving object database that can construct the optimal data structure. A database can calculate the optimal tree-based structure if all data sets of moving objects are simultaneously added to the database.

In the rest of this section, we explain the processes of constructing optimal MBRs.

1. At time t, the database temporally holds all positions $\boldsymbol{p}_0(t)$, $\ldots$, $\boldsymbol{p}_n(t)$ of moving objects that will be added to the database.
2. Positions are classified into each class by an average grouping method, a traditional hierarchical clustering method. As a result of the clustering process, each class has objects that are close to each other. In this clustering, the number of classes is indicated by a system administrator before string data. A database calculates MBRs for each class using the position of moving objects included in the class. These MBRs are used as leaf nodes of a tree-based data structure.
3. After the calculation of all leaf nodes, a database constructs each non-leaf MBR in the tree structure from the lower layer to the root.
4. Finally, a database calculates predictive MBRs from t to $t+i$ for every MBR in the tree.

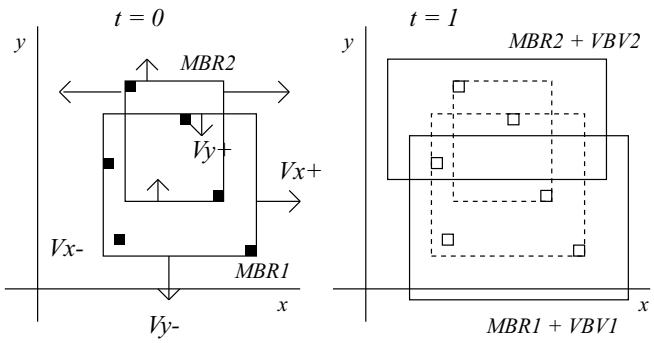

Fig. 4. Prediction on TPR-Tree

These processes enable databases to calculate effective MBRs that are smaller than MBRs calculated by traditional algorithms for tree construction. Since this technique can be applied to previous proposed tree structures, we will use it to compare our method with previous methods.

4 Evaluation

This section describes the results of experiments that compared our method with previous prediction techniques. In our evaluations, we compared two indicators that show the performance of prediction mechanisms: the reconstruction rate and the MBR validation rate. Reconstruction rate rec_{t+i} at time $t+i$ is given as $\sum_{\tau=t+1}^{t+i} r_\tau/(n_{MBR} * i)$, where n_{MBR} is the number of all original, non-predicted MBRs and r_τ is the number of reconstructed MBRs at time τ according to prediction errors. Whenever a prediction error occurs, a database must possibly reconstruct the tree structure. If no prediction errors occur from time t to $t+i$, the reconstruction rate becomes 0. The other indicator, MBR validation rate val_{t+i}, is given as σ_{t+i}/s_{t+1}, where s_τ is the area of an ideal MBR at τ and σ_τ is the area of a predicted MBR at τ. If no prediction errors occur at time $t+i$, val_{t+i} becomes 1 because the predicted MBR must equal the ideal MBR. Note that val_{t+i} is not larger than 1 since val_{t+i} is calculated after the reconstruction of MBRs, so that at least the size of the MBR equals the size of an ideal MBR.

4.1 Previous Works

Here, we mention previous techniques that predict the future positions of moving objects for constructing an effective data structure: TPR-Tree [9], TPR*-Tree [10], and STP-Tree [8]. To compare our technique with these previous works in our experiments, we briefly explain these schemes.

TPR-Tree In TPR-Tree and TPR∗-Tree, a database system predicts the future positions of objects using velocities from each axis. To predict a position, the

system calculates each maximum velocity of objects in an MBR by positive and negative x- and y-axes from time $t-m$ to t. These velocities are denoted as V_{x+}, V_{y+}, V_{x-}, and V_{y-}, as shown in Figure 4; for example, V_{x+} is the maximum velocity of all objects in an MBR by the positive x-axis during a period. When no object moves in a direction, for instance, no object moves toward the positive y-axis, as illustrated in Figure 4, value V_{y+} has a negative value. In TPR-Tree, the set of V_{x+}, V_{y+}, V_{x-}, and V_{y-} is called the Velocity Bounding Vector (VBV). A database calculates future MBRs from VBV; concretely, each corner point of a future MBR is given by the following equations:

$$\begin{aligned} P_{x-}(t+j) &= V_{x-} \times j \\ P_{y-}(t+j) &= V_{y-} \times j \\ P_{x+}(t+j) &= V_{x+} \times j \\ P_{y+}(t+j) &= V_{y+} \times j \end{aligned} \tag{5}$$

In this technique, the MBR validation rate is lower than other techniques because it only uses maximum velocity; however, its reconstruction rate is lower than others because the predicted MBR is always larger than the ideal MBR. When an object completely stops at a point or moves straight at the same velocity, in this technique a database must accurately predict the object's future point. Even if objects are moving randomly, the reconstruction rate is lower than others since the predicted MBR must be larger than the ideal MBR. On the other hand, when objects are moving randomly, the predicted MBR area tends to be much larger than the ideal MBR: in other words, the MBR validation rate becomes low. Similarly, when trajectories have much noise, MBR validation rates also become lower than other techniques.

In the original TPR-Tree, each MBR includes objects that are close to each other at time t. A database does not check overlaps between areas of constructed MBRs. In TPR*-Tree, a database checks for overlaps and reconstructs MBRs so that no MBR overlaps with other MBRs and the VBV of objects in an MBR is similar to each other, after constructing MBRs based on the TRP-tree technique. Since the prediction accuracy of the TPR*-Tree is possibly higher than the original TPR-Tree, in our experiment we compared our methods with it.

For evaluations, we apply our enhanced MBR construction methods, as mentioned in 3.4. After a database temporally stores all points of moving objects at each time, the database constructs optimal MBRs at the time by clustering techniques. For results, we use the same MBRs for evaluation in any tree structure by comparing reconstruction and MBR validation rates.

STP-Tree The prediction technique in STR-Tree uses a nonlinear predictive function represented by the past positions of an object. The essential idea is based on the calculation of approximate predictive functions using several past points through which an object has already passed. To calculate approximate function, STR-Tree uses SVM techniques, which are traditional signal processing techniques.

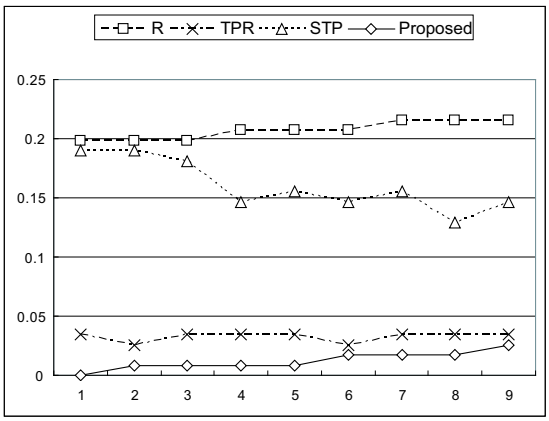

Fig. 5. Comparison results for reconstruction rate (data of moving vehicles)

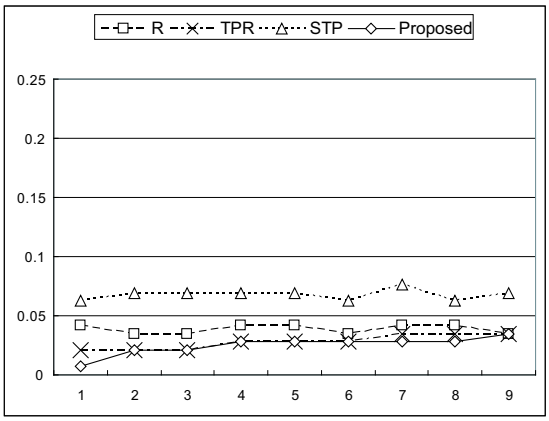

Fig. 6. Comparison results for reconstruction rate (data of walking visitors)

In STR-Tree, a database system makes a sequence of positions, such as $x(t-m), y(t-m), x(t-m+1), y(t-m+1), \ldots, x(t), and y(t)$ from time $t-m$ to t. We denote a sequence of position vectors from time $t-m$ to t as $\boldsymbol{k}(t)_m$; similarly, a vector sequence from $t-m-1$ to $t-1$ is denoted as $\boldsymbol{k}(t-1)_m$. For predictions, a database system makes $2m \times n$ matrix denoted as $\boldsymbol{K}(t)_{m,n}$ such that the top row of the matrix is given as $\boldsymbol{k}(t)_m$; also n-th row is given as $\boldsymbol{k}(t-n)_m (n < m)$. Another sequence, $\boldsymbol{x}(t)_n = \langle x(t-n), \ldots, x(t) \rangle$, is a sequence of an object's x-axis from time t to $t-n$; similarly, $\boldsymbol{y}(t)_n$ can be defined. Hence, we can calculate the approximate answer sequence of vector $\boldsymbol{w}_x = \langle wx_1, wx_2, \ldots, wx_{2m} \rangle$, $\boldsymbol{w}_y \langle wy_1, wy_2, \ldots, wy_{2m} \rangle$ in the following equations:

$$\boldsymbol{x}(t)^T = \boldsymbol{K}(t-1)_{m,n} \bullet \boldsymbol{w}_x^T \tag{6}$$

$$\boldsymbol{y}(t)^T = \boldsymbol{K}(t-1)_{m,n} \bullet \boldsymbol{w}_y^T.$$

An approximate answer can be calculated by Support Vector Machines (SVM). Matrix $\boldsymbol{w}_x$, $\boldsymbol{w}_y$ and vector $\boldsymbol{k}_m(t) = \langle x(t-m),\ y(t-m), x(t-m+1), y(t-m+1), \ldots, x(t), y(t)\rangle$, introduces position $\boldsymbol{p}(t+1) = \langle x(t+1), y(t+1)\rangle$ at $t+1$ as the following equation:

$$\begin{aligned} x(t+1) &= \boldsymbol{w}_x \bullet \boldsymbol{k}(t)_m^T \\ y(t+1) &= \boldsymbol{w}_y \bullet \boldsymbol{k}(t)_m^T. \end{aligned} \tag{7}$$

Positions $\boldsymbol{p}(t+2), \ldots$ after $t+2$, can be calculated by these recursive equations.

In this method, the system predicts the future point of an object based on the affine transformation on the coordinate system; the system accurately predicts future positions if an object moves in an arc, a straight line, or a sign curve. On the other hand, frequent turns by an object decrease prediction accuracy.

Note that in our experiments we also adapt our method to construct MBRs, as in the case of TPR-Trees.

4.2 Experiment Setting

We evaluated our proposed method with practical trajectory data, as mentioned in Section 2. For evaluation, we implemented three prediction and indexing methods on Windows XP and C♯ Language: TPR-Tree, STP-Tree, and our proposed method. In each method, we focused on two indicators, reconstruction rate and MBR validation rate, which use 10 points from time $t = -9$ to $t = 0$ to predict points from $t = 1$ to $t = 10$. If a method can completely construct future MBRs at time $t = n$, the reconstruction rate equals 0 at $t = n$. But if half of the objects exist outside of constructed MBRs, the reconstruction rate is 0.5. In other words, a low reconstruction rate means high prediction accuracy. The MBR validation rate is also an indicator of prediction accuracy, but a high MBR validation rate means high accuracy in contrast to the reconstruction rate because the MBR validation rate is calculated as the ratio of the predicted MBR area at time t to ideal MBR at time t. An ideal MBR is constructed such that the MBR completely includes real (not predicted) points of objects at time t, so the predicted MBR equals the ideal MBR if a system can completely predict the future positions of all objects. The rate of the complete predicted MBR becomes 1 since the predicted MBR, which is larger than the ideal MBR, will be reconstructed in a construction algorithm as an ideal MBR, as mentioned above.

Figures 5, 6, and 7 compare reconstruction rates, and Figures 8, 9, and 10 compare MBR validation rates. The horizontal axis of each figure denotes the past time ($t = 1$ to $t = 10$) from when a system predicted future positions. The time scale depends on the sampling time of each data. The vertical axis shows either reconstruction or the MBR validation rate for forty moving objects selected randomly from each data set.

The results of R-Tree [7] shown in the figures can be considered the rate without any prediction, since MBRs in R-Tree will always be reconstructed to

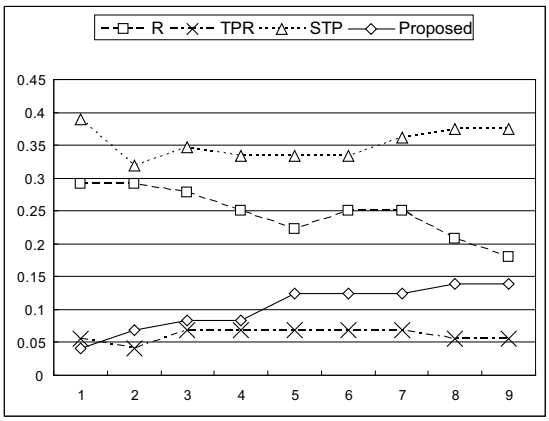

Fig. 7. Comparison results for reconstruction rate (data of sport motion)

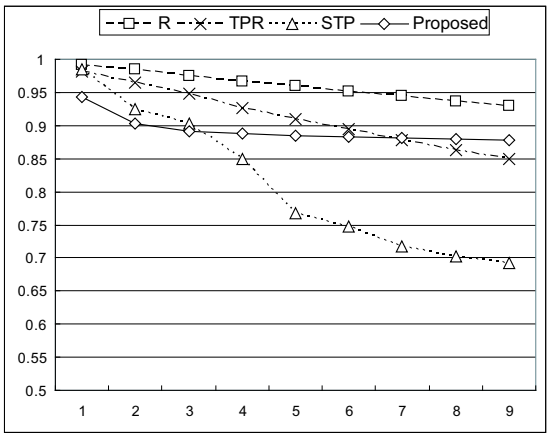

Fig. 8. Comparison results for MBR validation rate (data of moving vehicles)

an ideal MBR whenever an object moves outside of the original MBRs. In other words, if a method's value at a time is lower than the R-Tree value, the performance of the prediction method is worse than no prediction. On the other hand, the MBR validation rate tends to be higher than others because MBRs in R-Tree will frequently be reconstructed. We consider the MBR validation rate of R-Tree the optimal rate at each time.

4.3 Results

All our experimental results show the advantages of our proposed method over previous methods. In our experiment, STR-Tree, which is an improved TPR-Tree method, has disadvantages compared to other methods. The prediction function

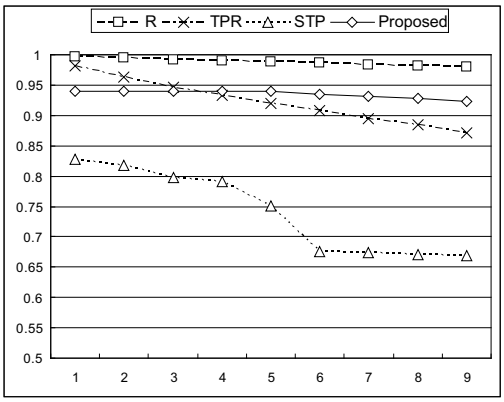

Fig. 9. Comparison results for MBR validation rate (data of walking visitors)

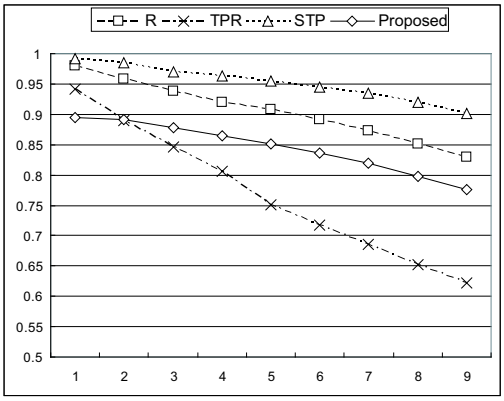

Fig. 10. Comparison results for MBR validation rate (data of sport motion)

of STR-Tree is too sensitive to noise; as a result the predicted points are often different from real points in practical data.

In the near future, the TPR-Tree results will show low reconstruction rates, but the rate will probably be lower than others in the distant future.

Each reconstruction rate from visitor data is lower than rates from the other two types of data because visitor trajectories have little noise, and many visitors stop at each exhibition for a long time. For sport data, many objects moving randomly decrease the STP-Tree and R-Tree performance. For all types of data, our proposed method maintains higher performance than other methods.

MBR validation rates of TPR-Tree and STP-Tree decrease more rapidly than our proposed method for predicted time. For the distant future, TPR-Tree constructs redundant predicted MBRs, which are much larger than ideal MBRs, although our proposed method can construct accurate predicted MBRs that match the real points of objects. About sport data, STP-Tree has an advantage

over our methods because objects moving randomly tend to become redundant MBRs in our methods. But our method's reconstruction rate is much better than STP-Tree, so on the whole, the performance of our method possibly has an advantage over STP-Tree.

We also examined the performance of an enhanced STP-Tree, which constructs MBRs larger than the original MBRs, the same as θ_p. In other words, we introduced θ_p into the STP-Trees to evaluate the effectiveness of θ_p. The introduction of θ_p certainly improved STP-Tree performance. But we basically only found slight improvement because the influence of the sensitive function is stronger than the improvement of θ_p. If we can reduce noise from the practical trajectory data, STR-Tree performance will possibly be improved more. Actually, it is difficult to reduce noise from several points of moving objects, so we conclude that our proposed method is better than other methods, even if θ_p is introduced to those other methods.

5 Conclusion

In this paper, we proposed a motion prediction method based on three motion patterns: staying, moving straight, and moving randomly to make predictive indexes for moving objects. Moreover, evaluation results showed the advantages of our methods in experiments that compared previous prediction techniques using practical trajectory data.

In our method, we suppose the trajectory data can be obtained accurately and completely; however, we should introduce a complementary method for missing trajectories. In the future, we will apply our method in application systems using trajectories and evaluate its performance in these systems with a complementary method. For applying practical application systems, we will also enhance our prediction technique based on geographic conditions; for example, when an object moves up a slope, its velocity probably decreases.

References

1. Lester, J., Choudhury, T., Borriello, G.: A practical approach to recognizing physical activities. In: International Conference on Pervasive Computing. (2006) 1–16
2. Hightower, J., Consolvo, S., LaMarca, A., Smith, I.E., Hughes, J.: Learning and recognizing the places we go. In: International Conference on Ubiquitous Computing (Ubicomp). (2005) 159–176
3. Wolfson, O., Sistla, P., Xu, B., Zhou, J., Chamberlain, S.: DOMINO: Databases fOr MovINg Objects tracking. In: SIGMOD'99 Conference Proceedings. (1999) 547–549
4. Mokhtar, H., Su, J., Ibarra, O.H.: On moving object queries. In: PODS2002 Symposium Proceedings. (2002) 188–198
5. Kollios, G., Gunopulos, D., Tsotras, V.J.: On indexing mobile objects. In: PODS'99 Symposium Proceedings. (1999) 261–272
6. Kollios, G., Tsotras, V.J., Gunopulos, D., Delis, A., Hadjieleftheriou, M.: Indexing animated objects using spatiotemporal access methods. IEEE Transactions on Knowledge and Data Engineering **13**(5) (2001) 758–777

7. Guttman, O.: R-trees: a dynamic index structure for spatial searching. In: SIGMOD'84 Conference Proceedings. (1984) 47–57
8. Tao, Y., Faloutsos, C., Papadias, D., Liu, B.: Prediction and indexing of moving objects with unknown motion patterns. In: SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM Press (2004) 611–622
9. Šaltenis, S., Jensen, C.S., Leutenegger, S.T., Lopez, M.A.: Indexing the positions of continuously moving objects. In: SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM Press (2000) 331–342
10. Tao, Y., Sun, J., Papadias, D.: Analysis of predictive spatio-temporal queries. ACM Trans. Database Syst. **28**(4) (2003) 295–336