Implementation and Evaluation of an Adaptive Neighborhood Information Retrieval System for Mobile Users

Yoshiharu Ishikawa† Yuichi Tsukamoto‡ Hiroyuki Kitagawa†
†*Institute of Information Sciences and Electronics*
‡*Graduate School of Systems and Information Engineering*
University of Tsukuba, Tsukuba, Ibaraki, 305-8573 Japan
{*ishikawa,kitagawa*}*@is.tsukuba.ac.jp, yuichi@kde.is.tsukuba.ac.jp*

Abstract

Rapid development and ongoing research activities on mobile devices, digital cartography, and global positioning systems (GPSs) have brought us a new type of software service—location-based services for moving objects (such as people with mobile devices and vehicles with car navigation systems). Realization of location-based services requires new technologies to provide appropriate neighborhood information to moving objects. A general approach to providing neighborhood information to moving objects is to retrieve objects in the neighborhood of a moving object with a spatial query that uses the traditional Euclidean distance. However, if we know the destination and the estimated route of a moving object, we would be able to provide more appropriate information to the object. Based on this idea, we have developed adaptive spatial query generation models that take the trajectory of a moving object into consideration to retrieve desired information. In this paper, we describe the design and implementation of the neighborhood information retrieval system based on the models and evaluate its effectiveness with experiments.

1. Introduction

Recent development of mobile technologies, GPS devices, and electronic cartography realized location-based information services that provide adequate information to *moving objects*, such as people with mobile devices and vehicles with car navigation systems, depending on the place where they are currently located. A typical approach to providing neighborhood information to a moving object is to retrieve *POIs* (POI stands for "point of interests" such as a shop, a hotel, and so on) around its current location using the Euclidean distance; for example, we can consider a query "select nearest five gas stations from here." However, this approach is not necessarily an appropriate solution when a moving object has its destination and we can obtain or calculate the estimated route to its destination. If we know the past and future trajectory of a moving object, we may be able to select more appropriate POIs for the object.

Figure 1 explains this idea. The figure shows the current position, the past trajectory, and the estimated future trajectory of a moving object. If we know such information, the appropriate search area to provide information to the moving object would be the shaded area in the figure.

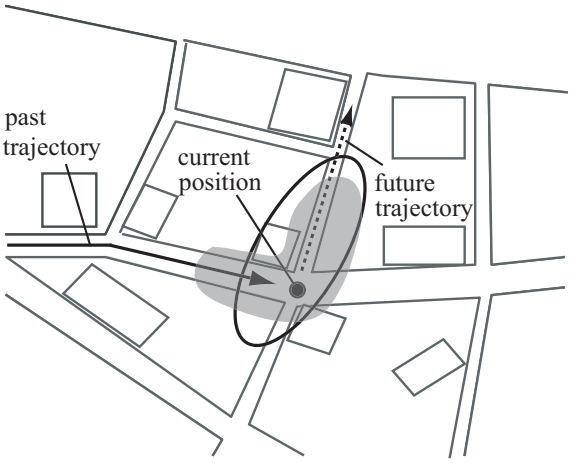

Figure 1. Providing neighborhood information to a moving object.

Based on this idea, we have proposed an approach that uses an ellipsoid region, as shown in Fig. 1, as an approximation of the shaded region [9]. An ellipsoid region which is issued to a spatial database as a spatial query is generated based on the query generation models on each movement of the object, and changes its shape according to the trajectory and the speed of the object. Based on the proposed query

generation models, we have implemented a prototype system of a neighborhood information retrieval system using ESRI's *ArcView GIS 2.3* and its extension package *Tracking Analyst* [5].

The rest of this paper is organized as follows. In the next section, we describe the related work. Section 3 introduces the query generation models. Section 4 shows the design and implementation of the prototype system. Section 5 presents the experiments conduced over the system and evaluates the pros and cons of the proposed models using the prototype system. Finally, Section 6 concludes the paper.

2. Related work

Due to the rapid development of mobile computing technologies, research on information providing services for mobile users have got its popularity (for example, see [4]). Support of moving objects also has become an important issue in the database research field [10, 11, 17]. Its main research areas include data modeling techniques for moving objects [7, 8] and indexing techniques to retrieve moving objects and/or the neighborhood of moving objects [1, 10, 12, 13, 14, 16, 18].

Most of the former approaches of information retrieval services for mobile users have utilized the conventional Euclidean distance to retrieve neighborhood POIs (point of interests). Although the Euclidean distance has benefits that it has clear semantics and efficient query evaluation methods are already available, it does not necessarily retrieve "neighborhood" POIs for moving objects. For example, if we retrieve nearest POIs using the Euclidean distance in Fig. 1, the result would include POIs which are located on the places where the user will not visit actually. An *ellipsoid distance* used in our approach can change its shape and rotation based on the specified parameters and enables us to retrieve POIs which are located along the trajectory of a moving object.

Ellipsoid distances have been used in various fields including pattern recognition and statistics, and also used in multimedia information retrieval [6]. Since we can adjust the shape and the rotation of the query region of an ellipsoid distance by setting appropriate parameters, we can generate an adequate query depending on the location and the situation of a moving object. Spatial queries based on an ellipsoid distance, called *ellipsoid queries*, also have a benefit that devised techniques already exist for their efficient evaluation [2, 3, 15]. It means that we can achieve efficient retrieval even for large spatial databases with the help of spatial indexes. In [9], we have already described the strategy to enable efficient evaluation of an ellipsoid query in our situation and the performance evaluation results of the proposed strategy.

3. Query generation models

In this section, we introduce the ellipsoid query generation models which consider the past and future trajectories of a moving object to estimate an appropriate query for the object. The detail of the models can be found in [9].

3.1. Representation of moving positions

Figure 2 represents the trajectory of a moving object that started $\boldsymbol{x}_1$ when $t = 1$. The location of the object at the current time $t = \tau$ is $\boldsymbol{x}_\tau$, and the estimated arriving time at the destination $\boldsymbol{x}_{\tau+\tau'}$ is $t = \tau + \tau'$. Note that the half of the trajectory shown in the figure is the estimated one. The remaining location vectors $\boldsymbol{x}_2, \ldots, \boldsymbol{x}_{\tau-1}, \boldsymbol{x}_{\tau+1}, \ldots, \boldsymbol{x}_{\tau+\tau'}$ are obtained by sampling the trajectory on every unit time. The basic idea behind our approach is that the movement status of a moving object at the current time ($t = \tau$) can be represent by this set of location vectors. The idea is formalized in the following subsection.

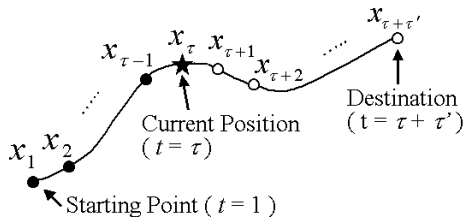

Figure 2. Modeling of route information.

According to future trajectories, we assume that a route estimation facility, which computes a new estimated trajectory on each detected movement of a moving object, is available. This is a natural assumption when we consider the current mobile technology and location-based services such as car navigation systems. As described later, since our approach only requires recent past location vectors and near future ones to generate a query, the route estimation system does not necessarily need to compute the full trajectory of a moving object.

3.2. Decay model of influence values

In [9], we have proposed the *decay model of influence values* of location vectors which is based on the idea that the influence (importance) of a location vector exponentially decreases toward past and future from the current position of the moving object. The idea can be summarized as follows:

- We set the highest importance on the location at the time σ ($0 \leq \sigma \leq \tau'$) unit times after from the cur-

rent time. Namely, $\boldsymbol{x}_{\tau+\sigma}$, the location where the object will move after σ unit times later, has the highest influence value. The idea behind this is that a mobile user usually has interests on the place where he or she will arrive in near future.

- Influence values decrease exponentially toward past and future based on the parameters μ and ν ($0 < \mu < 1, 0 < \nu < 1$), respectively.

We represent the influence value of the location vector $\boldsymbol{x}_t$ as

$$\alpha(t) = \begin{cases} \mu^{\tau+\sigma-t} & (t = 1, \ldots, \tau+\sigma) \\ \nu^{t-\tau-\sigma} & (t = \tau+\sigma, \ldots, \tau+\tau'). \end{cases} \quad (1)$$

The shape of the formula is shown in Fig. 3. By setting μ, ν, and σ appropriately, we can adapt our query generation models according to the mobile user's situation and preference. The query generation models shown in the next subsection are based on this influence decay model.

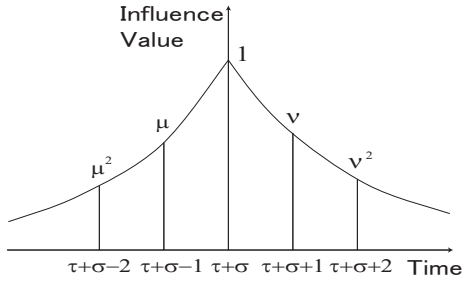

Figure 3. Exponential decay model of influence.

3.3. Query generation methods

Generally speaking, a spatial query can be specified in terms of the following three factors:

1. query center ($\boldsymbol{q}$)
2. distance function ($\mathcal{D}$)
3. query task

As described later, our prototype system supports two query center derivation methods, three types of distance functions, and two types of query tasks (range queries and k-nearest neighbor queries). In the following subsections, we describe the derivation methods for query centers and distance functions.

3.3.1 Derivation models for query centers

Two models to derive query centers are shown below.

1) Use of the current target position (model cur) This model is based on a simple approach and does not consider past and future location vectors; it simply uses the current target position $\boldsymbol{x}_{\tau+\sigma}$, where the moving object will arrive after σ unit times later.

2) Weighted average (model avg) This model considers the past and future trajectories and takes the weighted average of the location vectors using influence values described before. The query center is determined as

$$\boldsymbol{q} = \bar{\boldsymbol{x}} = \frac{\sum_{t=1}^{\tau+\tau'} \alpha(t)\boldsymbol{x}_t}{\sum_{t=1}^{\tau+\tau'} \alpha(t)}. \quad (2)$$

Since this model derive the query center by a weighted average, a location vector has a large effect when it is temporally nearer to the target point $\boldsymbol{x}_{\tau+\sigma}$. Since we can obtain the derivation model **cur** when we set parameters as $\mu = 0$ and $\nu = 0$, this approach can be considered as a generalized version of **cur**.

3.3.2 Derivation models of distance functions

For distance functions, we have proposed the following three models.

1) The Euclidean distance (model EU) The simplest approach is the use of the standard Euclidean distance. In this approach, after the derivation of the query center $\boldsymbol{q}$, the distance is calculated as follows:

$$\mathcal{D}(\boldsymbol{x}, \boldsymbol{q}) = \sqrt{(\boldsymbol{x}-\boldsymbol{q})^t(\boldsymbol{x}-\boldsymbol{q})}. \quad (3)$$

Although the Euclidean distance does not utilize trajectory information, it has benefits of clear semantics and efficient computation.

2) Ellipsoid distance-based approach (model OV) We first introduce the notion of an ellipsoid distance. It is defined as

$$\mathcal{D}(\boldsymbol{x}, \boldsymbol{q}) = \sqrt{(\boldsymbol{x}-\boldsymbol{q})^t\mathbf{A}(\boldsymbol{x}-\boldsymbol{q})}, \quad (4)$$

where $\mathbf{A}$ is a $d \times d$ *symmetric positive definite matrix* ($\mathbf{A}^T = \mathbf{A}$ and $\boldsymbol{x}^T\mathbf{A}\boldsymbol{x} > 0$ for any $\boldsymbol{x} \neq 0$) and d is the number of dimensions (in our case $d = 2$). Since the distance defined above has isosurfaces of ellipsoid shapes, it is called an *ellipsoid distance*. The matrix $\mathbf{A}$ is called the *distance matrix*. Note that if $\mathbf{A}$ is a unit matrix, the ellipsoid distance is reduced to the Euclidean distance.

An ellipsoid distance has a benefit that we can tune the distance shape by setting the distance matrix **A** adequately depending on the situation. A distance function derivation model **OV** (it means an "oval") considers influence values $\alpha(t)$, shown in Subsection 3.2, to derive an appropriate **A**. As discussed in [9], the following distance matrix $\mathbf{A}_{\rm opt}$ that minimizes the summation (penalty) gives the "optimal" ellipsoid shape in terms of the given location vectors:

$$\mathbf{A}_{\rm opt} = \underset{\mathbf{A}}{\rm argmin} \sum_{t=1}^{\tau+\tau'} \alpha(t)(\boldsymbol{x}_t - \boldsymbol{q})^T \mathbf{A}(\boldsymbol{x}_t - \boldsymbol{q}). \quad (5)$$

The matrix $\mathbf{A}_{\rm opt}$ can be derived by setting an appropriate constraint $\det(\mathbf{A}) = 1$ on **A** as follows:

$$\mathbf{A}_{\rm opt} = \det(\mathbf{C})^{\frac{1}{d}} \mathbf{C}^{-1}, \quad (6)$$

where $\det(\mathbf{A})$ is the determinant of **A** and **C** is the following weighted covariance matrix:

$$\mathbf{C} = \sum_{t=1}^{\tau+\tau'} \alpha(t)(\boldsymbol{x}_t - \boldsymbol{q})(\boldsymbol{x}_t - \boldsymbol{q})^T. \quad (7)$$

3) Hybrid approach (HB) Although the above model **OV** has a benefit that it takes past and future trajectories into consideration, it has a problem that it is not *robust* for specific types of movement patterns. For example, if a moving object continually moves on a straight line or stops at a single point for a long period, the covariance matrix shown in Eq. (7) approaches to a *nonsingular matrix* (namely, $\det(\mathbf{C}) \simeq 0$); it brings problems such that the isosurface of an ellipsoid distance has a quite narrow shape. Its reason is that the model **OV** uses the spatial distribution of the past and future trajectories which is actually represented by a set of location vectors. The aim of the hybrid approach called **HB** is to alleviate this behavior and to make **OV** more robust.

The idea of the hybrid model is simple; it is based on the *regularization* technique, which is often used in statistics. This model first computes the matrix $\mathbf{C}^+$ as

$$\mathbf{C}^+ = \lambda \frac{\mathbf{C}}{\|\mathbf{C}\|} + (1-\lambda)\frac{\mathbf{I}}{\|\mathbf{I}\|}, \quad (8)$$

where $\|\mathbf{C}\|$ represents the Frobenius norm of **C** and **I** represents the unit matrix. The distance matrix **A** is given by

$$\mathbf{A} = (\det(\mathbf{C}^+))^{\frac{1}{d}} (\mathbf{C}^+)^{-1}. \quad (9)$$

The parameter λ ($0 \leq \lambda \leq 1$) is called the *hybrid parameter* and determines how to blend the two models **EU** and **OV**. When $0 \leq \lambda < 1$ is satisfied, it is assured that $\mathbf{C}^+$ is a nonsingular matrix. As special cases, when $\lambda = 0$ and 1, it reduces to the model **EU** and **OV**, respectively. Therefore, this model is a generalized version of the two models.

4. Design and implementation of the prototype system

In this section, we describe the design and implementation of the prototype system which is based on the query generation models described in Section 3.

4.1. System overview

The architecture of the prototype system is shown in Fig. 4. The prototype system is implemented using ESRI's *ArcView GIS 2.3*, a commercial GIS software package [5]. Also, *Tracking Analyst*, an extension package of ArcView is used to implement the dynamic behavior of the system. The system is developed using the *Avenue* scripting language, the basic scripting language for ArcView.

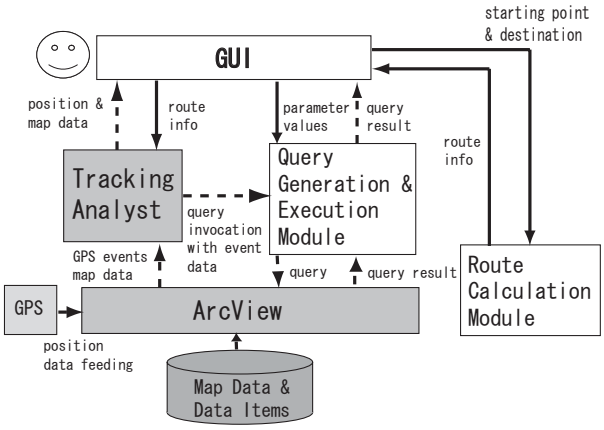

Figure 4. System architecture.

Figure 5 shows the user interface of the system. It first receives the starting point and the destination from a user. The route calculation module receives these parameters and calculates an estimated route to the destination. Tracking Analyst manages the estimated route for query processing. If a new location of the moving object is acquired from the GPS module, it issues a request to calculate a new estimated route. The user interface also receives the specification of parameter values from the user. Such values include the query task (range query or k-nearest neighbor query), the weighting parameters for past and future (μ and ν), the lookahead parameter (σ), the query center derivation model (**cur** or **avg**), the distance function derivation model (**EU**, **OV**, or **HB**), and the hybrid parameter λ in the case when the **HB** model is selected.

When the GPS module reports the current location of the moving object, the system generates a new neighborhood query, which consists of a query center $\boldsymbol{q}$, a distance

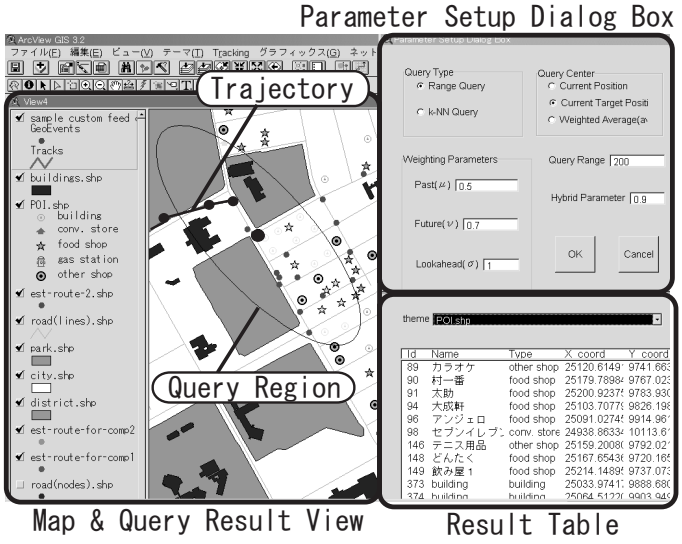

Figure 5. Graphical user interface.

function $\mathcal{D}$, and the query task, and executes it over the ArcView system, then the result is displayed on the GUI. The retrieved POIs are highlighted on the GUI and attribute values (such as a shop name and a shop type) of each POI is shown as a record in the displayed table.

4.2. Details of the modules

The modules and their relationships are shown in Fig. 4. The solid lines represent the data and the control flows required to set up initial parameters. The dotted lines represent the flow on the retrieval time. The roles of the modules are explained below.

1) ArcView The ArcView system supports the basic GIS facility; it manages map data and POI items, and retrieves them when the query generation and execution module sends it a request. Also, it has a function to send Tracking Analyst the detected current location of the moving object, which is obtained from the GPS module.

2) Tracking Analyst Using this extension package of ArcView, it is possible to display and analyze the obtained event-based data in a real-time manner. In the implementation the prototype system, we have extended the basic functionality of Tracking Analyst with programs written in the Avenue scripting language. Based on this extension, a new query invocation request is issued from Tracking Analyst to the query generation and execution module when it detects the movement event of the target moving object.

In the experiments, however, we did not use a real GPS system since experiments under equal conditions are quite difficult in real situations. Instead, we have implemented a Java program which simulates the role of a GPS system and sends simulated GPS signals to the input port of the computer. The prototype retrieval system considers these signals as if they are actual GPS signals.

3) Route calculation module The current route calculation module supports the route estimation functionality only for the part of Tsukuba city, Japan. The reason is that it is suffice for our experiments and that the route estimation is not our research focus. We have a plan to incorporate *Network Analyst*, an another ArcView extension package, to our system for dynamic route estimation.

4) Query generation and execution module This module implements the core part of our system and written in the Avenue language. The main role of the module is query generation and execution. When a user movement is notified from Tracking Analyst, it generates an appropriate query based on the specified parameters from the user, and sends it to the underlying ArcView system. The received POIs are ranked in the module based on the ellipsoid distance. Finally, the ranked POIs are returned to the GUI module.

5. Experimental evaluation

As the criteria to evaluate our system and the proposed models, we can consider two factors: *effectiveness* (the quality of query results) and *efficiency*. This section shows the experimental results for the effectiveness factor. According to the efficiency factor, we briefly mention it in Subsection 5.5.

5.1. Setting up the experiment

In this experiment, we use k-nearest neighbor queries as query tasks. Namely, the system retrieves top-k items using a given distance function $\mathcal{D}$ and a query center $\boldsymbol{q}$. We set the parameter as $k = 5$ throughout the experiments.

As the target data, we use the map and POIs in Tsukuba city, Japan. The POI set consists of about 200 items including gas stations, shops, restaurants, office buildings and so on. In the experiments, we use the driving route shown in Fig. 6. The route is obtained from an actual car driving in the city. It takes about five minutes for the five kilometer route from the starting point to the destination. From this driving data, we calculate the location vectors by sampling the data with every five seconds. Therefore, the unit time in the experiments is five seconds.

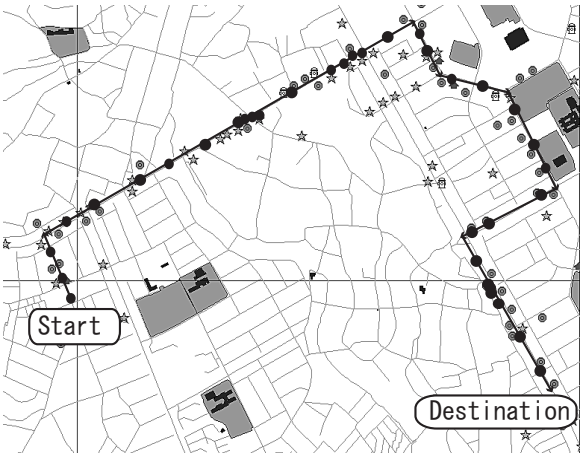

Figure 6. Driving route.

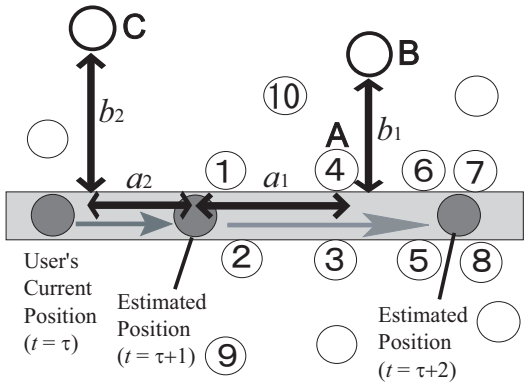

Figure 7. Ranking method for the ideal retrieval result.

5.2. Construction of the ideal retrieval result set

To compare the retrieval effectiveness of the proposed query generation models, we first construct the *ideal retrieval result set* which consist of the ideal retrieval result for each query point. By the term "ideal retrieval result", we mean the optimal POI ranking which reflects the psychological distance of a mobile user. If the ideal retrieval result is given to a query point, we can evaluate the effectiveness of a query generation method by comparing the ideal ranking with the ranking generated by the system. However, it is quite difficult to obtain such a ranking since different mobile users usually have different preferences. Therefore, we propose a simple user model which simulates the behavior of a typical mobile user who

- puts the highest importance weight where he or she will arrive after a unit time later (five seconds later in the experiments),
- considers the future trajectory more important than the past trajectory, and
- regards POIs along the road more important than POIs which are apart from the road.

We explain this idea using Fig. 7. The shaded circles represent the positions of a mobile user at $t = \tau$ (the current time), $t = \tau + 1$, and $t = \tau + 2$, respectively. The white circles are POIs. Based on the assumption described above, our user puts the highest importance weight on the position at $t = \tau + 1$.

Then we calculate the "distance function" of the assumed user as below. We denote the *user's distance* to POI x from the user's current position by $d_{\mathrm{u}}(x)$. It is defined as follows:

- For a POI that is along the road, we simply use the Euclidean distance to the POI. For example, we compute the user's distance to the POI "A" shown in the figure as $d_{\mathrm{u}}(\mathrm{A}) = a_1$.
- For a POI that locates on the forward position but is apart from the road, we add a unit time penalty, considering the time required to turn the road, to the city block distance between the current position and the POI. For the POI "B" in the figure, the distance is calculated as

$$d_{\mathrm{u}}(B) = a_1 + b_1 + udd,$$

  where udd, the *unit driving distance*, is calculated as an average distance that the mobile user can move within a unit time.
- For a POI that locates on the backward position of the user, we add two unit driving distances to the city block distance between the current position and the POI. For instance, the user's distance to the point "C" is computed as

$$d_{\mathrm{u}}(C) = a_2 + b_2 + 2 \times udd.$$

Each number shown in the POIs in Fig. 7 represents the rank of the POI based on the user's distance. In this figure, the ranks are calculated until top-10 nearest POIs.

5.3. Evaluation measure

To compare the ideal retrieval result calculated as described in the previous subsection and the actual retrieval result, we propose an evaluation measure. It is a modified version of the precision measure usually used in information retrieval.

We assume that POIs within top-p ranks in the ideal retrieval result set are the *relevant POIs*. We evaluate the retrieval effectiveness of a query generation model by calculating how many POIs within the top-k retrieved POIs are included in the relevant POIs. The calculation method is formally defined as follows. First, let $Res(k)$ be the top-k POIs in the actual retrieval result set and let $Ans(p)$ be the top-p POIs in the ideal result set. The evaluation measure $Precision(k,\ p)$ is given by the following formula:

$$\begin{aligned} Precision(k,p) &= \frac{|Res(k) \cap Ans(p)|}{|Res(k)|} \\ &= \frac{|Res(k) \cap Ans(p)|}{k}. \end{aligned} \quad (10)$$

In the following experiments, we use the settings $k = 5$ and $p = 5$ and 10.

5.4. Experimental results

In this subsection, we present the results of the evaluation experiments.

1) Comparison of EU and HB In the first experiment, we compare the Euclidean distance-based approach (**EU**) and the ellipsoid distance-based approach (**OV** and **HB**). Since **OV**, the simple ellipsoid distance-based approach, is a special case of the hybrid approach **HB**, we select **HB** as the representative one in this experiment. We use a typical parameter setting $\mu = 0.4$, $\nu = 0.8$, $\sigma = 2$, and $\lambda = 0.9$ for **HB**.

Figure 8 shows precision scores of **EU** and **HB** for the driving route shown in Fig. 6. The horizontal axis is the transition time when we set the start time as $t = 0$ (the unit time is five seconds). The vertical axis represents the precision score. Although **EU** is better in some points for $Precision(5,\ 5)$, which considers top-5 of the ideal result set as relevant, the overall precision is worse than **HB**. Our analysis said that at the point where **HB** is worse than **EU**, there is a tendency that the actual distribution of POIs is sparse and/or not along the road. For $Precision(5, 10)$, that regards top-10 of the ideal result set entries as relevant, **HB** is more better than **EU**. Based on this experiment, it is shown that the ellipsoid distance-based approach is generally better than the simple Euclidean distance-based approach at least we use appropriate parameter settings.

2) Comparison of query center derivation methods In the remaining experiments, we focus on the properties of ellipsoid distance-based approach. First, we compare two types of query center derivation methods **cur**, that focuses on the current target point, and **avg**, which derives the query point as the weighted average of the trajectory points. Figure 9 shows their precision scores. In the experiment, we

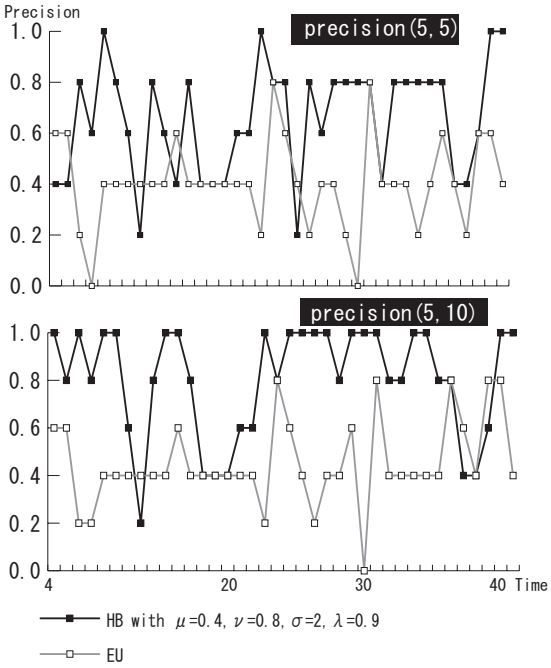

Figure 8. Comparison of EU and HB.

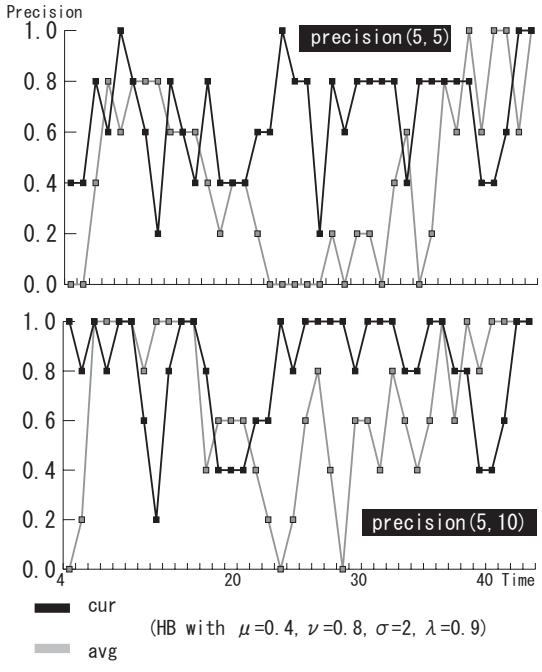

Figure 9. Comparison of cur and avg.

have used **HB** with parameters $\mu = 0.4$, $\nu = 0.8$, and $\lambda = 0.9$. The σ-value is set as $\sigma = 2$.

As shown in the figure, **cur** is better than **avg** for the most of the ranges. The **avg** method has large differences between good and bad cases. The periods when **avg** show poor performance ($t = 4$ to 5 and $t = 25$ to 30) correspond to the situations when the moving object turns corners. As shown in Fig. 10, the **avg** method has a tendency to select the inside of a corner as a query center so that it will lose POIs on the outside of the corner. From this result, we can say that **cur** generally returns stable results. But note that one of the reason of this behavior is that we assume a typical user who has high interests on POIs along the road when we construct the ideal retrieval result set. Since **cur** always generates query centers located on the road, it should be more effective than **avg** in this experiment. Use of other user models might change the behavior of two models.

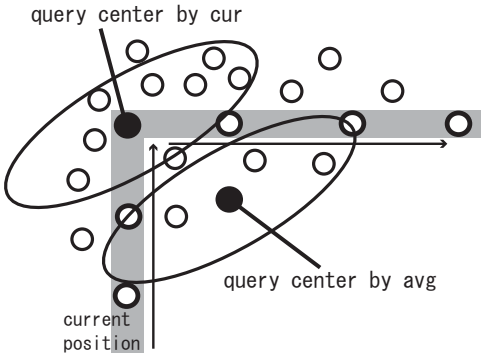

Figure 10. Differences between cur and avg.

3) Comparison between different past and future parameter settings In this experiment, we take different combinations of past and future parameters μ and ν, then compare their results. In this experiment, we use the combination of **HB** and **cur**. Figure 11 shows precision scores for the following three parameter combinations:

- a standard setting ($\mu = 0.4$ and $\nu = 0.8$),
- high weights on future ($\mu = 0.1$ and $\nu = 0.9$), and
- even weights ($\mu = 0.5$ and $\nu = 0.5$).

The standard weighting ($\mu = 0.4$ and $\nu = 0.8$) gives the best result on average. The high "future" weighting ($\mu = 0.1$ and $\nu = 0.9$) shows large differences between good and bad scores. Compared to experiment 1 and 2, different parameter settings do not bring drastic differences. One of the reason is that we have generated query centers with **cur**, the model which does not consider the weighting parameters so that three methods use the same query centers.

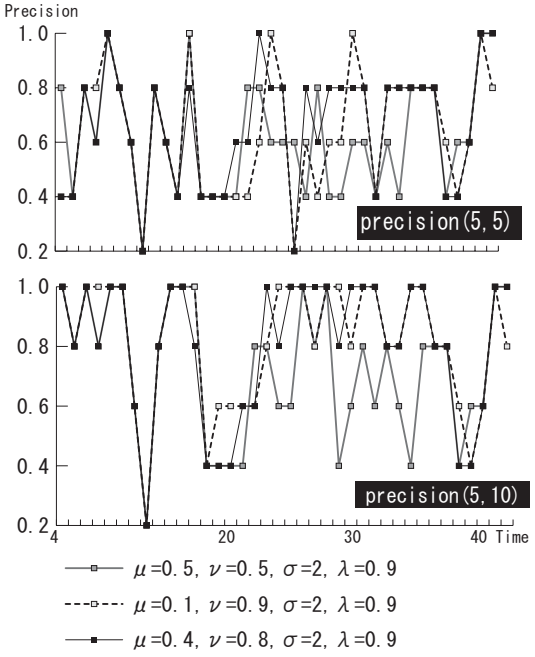

Figure 11. Comparison of different past and future weightings.

4) Comparison of different hybrid parameter settings In this experiment, we examine the role of the hybrid parameter λ for **HB**. We use **cur** as the query center derivation method and set parameters as $\mu = 0.4$ and $\nu = 0.8$. As described before, when the hybrid parameter λ becomes small the distance approaches to the Euclidean distance. Oppositely, if we use a large λ value ($\simeq 1$), it is assumed to behave like **OV**.

Figure 12 compares three parameter settings:

- a standard setting $\lambda = 0.9$,
- an "oval-like" setting $\lambda = 0.98$, and
- a "Euclidean-like" setting $\lambda = 0.7$.

As we can see in the figure, the changes of λ values do not show large differences in terms of the averaged scores, and the standard setting $\lambda = 0.9$ is the most effective one. When we use the large hybrid parameter value $\lambda = 0.98$, the differences between good and bad scores become large; the reason would be that the shapes of ellipsoids become quite narrow and lose target POIs. The result says that the hybrid approach **HB** is generally better than the simple ellipsoid distance-based approach **OV** when we use an appropriate λ-value setting; in this experiment, $\lambda = 0.7$ is the most effective one.

The overall results throughout the experiments are summarized as follows. The ellipsoid distance-based approach

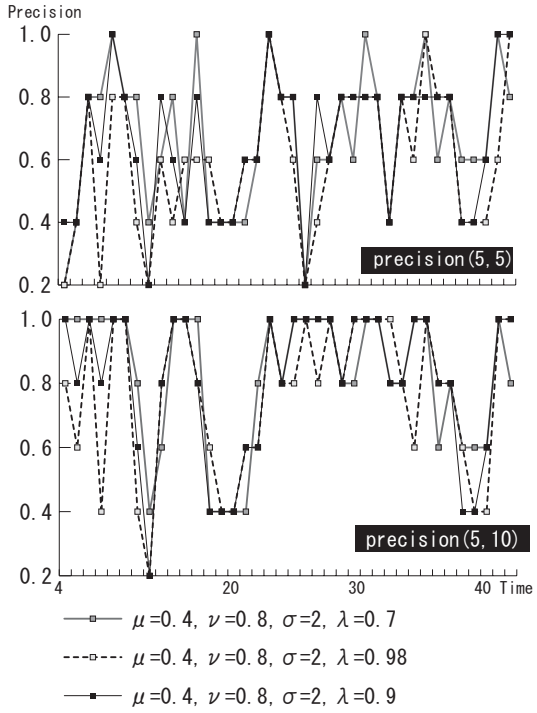

Figure 12. Comparison of different hybrid parameter settings.

generally returns a better retrieval result than **EU**. According to the query center derivation methods, **cur** is better than **avg** in our experiment setting. In terms of weighting, high "future" weighting compared to "past" weighting gives better results. But if the future weight is too large, it may bring bad effects. In conclusion, the hybrid model **HB** with an appropriate λ-value setting is the most effective approach in terms of precision scores.

5.5. Efficiency of the system

In this subsection, we briefly mention the retrieval efficiency of the system. Generally speaking, the retrieval cost of a query in our system mainly consists of 1) query generation, 2) query processing, and 3) query result presentation.

For efficient query processing, after the generation of an ellipsoid query, our system first generates a rectangular-shaped bounding query to retrieve candidate objects that may satisfy the given ellipsoid query. The rectangular-shaped query is submitted to the ArcView system and executed in it, then the resulting candidate objects are returned. Finally, the returned candidate objects are ranked by the ellipsoid distance to compute the final result. Since the first filtering query can reduce the number of candidate objects drastically, it is an efficient query processing approach. We omit the detail of the approach because we have already reported the evaluation result in [9].

In the above experiments that simulate the actual system usage with the unit time setting of five seconds, the prototype system actually computes each cycle consisting of query generation, query execution, and result display on the GUI within this unit time. The detailed analysis of the behavior of program execution has revealed that the execution overhead of the Avenue scripting language and the result display on the GUI have large percentage of the elapsed time. Therefore, the cost of query generation and processing is considered to be rather small.

6. Conclusions

In this paper, we have described the design and implementation of a neighborhood information retrieval system for moving objects. Its main feature is that it considers the route and the speed of a moving object and generates an appropriate spatial query at each movement of the object. The key technique is the use of the ellipsoid distances, instead of the traditional Euclidean distance. The ellipsoid distance has a benefit that we can modify the shape of a query by specifying appropriate parameters. We have described the query generation models and they are implemented in the prototype system.

In the experiments, we have focused on the query effectiveness issue and compared the precision scores of the query generation models based on the real user movement data and a POI dataset. Also, we have examined the behaviors of different parameter settings in our query generation models. Based on the experiments, we have shown that the hybrid approach which is based on the ellipsoid distance, but blends the Euclidean feature with it, has a good overall performance in general situations. The future work includes the incorporation Network Analyst, an extension package of ArcView, to enhance the system with a route estimation facility.

Acknowledgments

This research is partly supported by the Grant-in-Aid for Scientific Research on Priority Areas (14019009) from the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan, the Grant-in-Aid for Young Scientists (B) (14780316) and the Grant-in-Aid for Scientific Research (B) (12480067) from Japan Society for the Promotion of Science (JSPS), Japan, and the Research Grant from SECOM Science and Technology Foundation.

References

[1] P. K. Agarwal, L. Arge, and J. Erickson. Indexing moving points. In *Proc. ACM PODS*, pages 175–186, 2000.

[2] M. Ankerst, B. Braunmüller, H.-P. Kriegel, and T. Seidl. Improving adaptable similarity query processing using approximations. In *Proc. VLDB*, pages 206–217, 1998.

[3] M. Ankerst, H.-P. Kriegel, and T. Seidl. A multistep approach for shape similarity search in image databases. *IEEE TKDE*, 10(6):996–1004, 1998.

[4] N. Davies, K. Cheverst, K. Mitchell, and A. Efrat. Using and determining location in a context-sensitive tour guide. *IEEE Computer*, 34(8):35–41, 2001.

[5] ESRI Home Page. `http://www.esri.com/`.

[6] C. Faloutsos. *Searching Multimedia Databases by Content*. Kluwer Academic Publishers, 1996.

[7] L. Forlizzi, R. H. Güting, E. Nardelli, and M. Schneider. A data model and data structures for moving objects databases. In *Proc. ACM SIGMOD*, pages 319–330, 2000.

[8] R. H. Güting, M. H. Böhlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, and M. Vazirgiannis. A foundation for representing and querying moving objects. *ACM TODS*, 25(1):1–42, 2000.

[9] Y. Ishikawa, H. Kitagawa, and T. Kawashima. Continual neighborhood tracking for moving objects using adaptive distances. In *Proc. International Database Engineering and Application Symposium (IDEAS'02)*, pages 54–63, Edmonton, Canada, 2002.

[10] C. S. Jensen (ed.). Special issue on indexing of moving objects. *Bulletin of the Technical Committee on Data Engineering*, 25(2), June 2002.

[11] C. S. Jensen (ed.). Special issue on infrastructure for research in spatio-temporal query processing. *Bulletin of the Technical Committee on Data Engineering*, 26(2), June 2003.

[12] G. Kollios, D. Gunopulos, and V. T. Tsotras. On indexing mobile objects. In *Proc. ACM PODS*, pages 261–272, 1999.

[13] D. Pfoser, C. S. Jensen, and Y. Theodoridis. Novel approaches in query processing for moving objects. In *Proc. VLDB*, pages 395–406, 2000.

[14] S. Šaltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez. Indexing the positions of continuously moving objects. In *Proc. ACM SIGMOD*, pages 331–342, 2000.

[15] T. Seidl and H.-P. Kriegel. Efficient user-adaptable similarity search in large multimedia databases. In *Proc. VLDB*, pages 506–515, 1997.

[16] Y. Tao and D. Papadias. Spatial queries in dynamic environments. *ACM TODS*, 28(2):101–139, June 2003.

[17] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang. Moving objects databases: Issues and solutions. In *Proc. 10th SSDBM*, pages 111–122, Capri, Italy, 1998.

[18] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. L. Lee. Location-based spatial queries. In *Proc. ACM SIGMOD*, pages 443–454, June 2003.