Expert Systems with Applications 38 (2011) 15075–15083

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A Kalman filter updating method for the indoor moving object database

Jaegeol Yim [a], Jaehun Joo [b,*], Chansik Park [c]

[a] Department of Computer Science, Dongguk University, Gyeongju-si, Gyeongsangbuk-do 780-714, South Korea
[b] Department of Information Management, Dongguk University, Gyeongju-si, Gyeongsangbuk-do 780-714, South Korea
[c] School of Electrical and Computer Engineering, Chungbuk National University, South Korea

ARTICLE INFO

Keywords:
Location-based service
Moving object database
Indoor positioning
Kalman filter
Updating strategy

ABSTRACT

A moving object database (MODB), a database representing information on moving objects, has many uses in a wide range of applications, such as the digital battlefield and transportation systems. In the transportation system, an MODB processes queries such as "How long should I wait until the next bus arrives here?" Therefore, location information on moving objects reflects the most important data the MODB has to manipulate. Most moving objects are equipped with a GPS (Global Positioning System) unit that sends location information to the MODB. However, GPS signals are usually very weak inside enclosed structures; thus, locating indoor moving objects requires more than the GPS. In this regard, indoor positioning for location-based services (LBSs) has been an important research topic for the last decade. There are many other differences between indoor and outdoor MODBs. For examples, the area where the indoor moving objects are moving around is much smaller than where the outdoor moving objects are moving around, and the speed of indoor moving objects is much slower than that of outdoor ones. Therefore, the indoor moving object database (IMODB) should be studied separately from the outdoor MODB or the MODB.

One of the most important problems that the MODB has to solve is the updating problem. In this regard, this paper proposes an updating method of IMODBs for location-based services. Our method applies the Kalman filter to the most recently collected series of measured positions to estimate the moving object's position and velocity at the last moment of the series of the measurements and extrapolates the current position with the estimated position and velocity. If the difference between the extrapolated current position and the measured current position is less than the threshold, that is, if the two positions are close, we skip updating the IMODB.

When the IMODB requires information on the moving object's position at a certain moment T, it applies the Kalman filter to the series of the recorded measurements at the moments before T and extrapolates the position at T with the Kalman filter in the same manner as the updating process described earlier. To verify the efficiency of our updating method, we applied our method to a series of measured positions obtained by employing the fingerprinting indoor positioning method while we walked through the test bed. We then analyzed the test results to calculate savings of communication cost and error.

1. Introduction

A moving object database (MODB) is a database representing information on moving objects, particularly their locations. The purpose of the MODB is to provide answers to various spatiotemporal information queries such as the following: "Where am I?" "Who is the closest friend to me?" "How far is he?" "When and where did I meet Kim?" A location-based service (LBS), such as the fleet management system for a taxi company or a logistics company, cannot be realized without the availability of spatiotemporal information. Therefore, MODB techniques have been developed in the LBS field.

For most MODB applications, the moving object is equipped with a GPS (Global Positioning System) unit that sends information on the object's location, time, and velocity, among others, to the MODB. However, GPS signals are usually unavailable inside enclosed structures; thus, MODB techniques used for outdoor LBSs cannot be directly applied to indoor LBS systems. An indoor moving object can be a mini-notebook computer, a PDA, or a smart phone carried by a pedestrian. The physical care covered by indoor moving objects is much smaller than that by outdoor ones, and the speed of indoor moving objects is much slower than that of outdoor ones.

Indoor LBSs are as much more useful than outdoor LBSs. Indoor LBSs may include an electronic museum guide, a shopping

* Corresponding author. Tel.: +82 54 770 2346; fax: +82 54 770 2310.
E-mail addresses: yim@dongguk.ac.kr (J. Yim), givej@dongguk.ac.kr (J. Joo), chansp@chungbuk.ac.kr (C. Park).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.05.037

concierge at a department store, asset tracking, VoIP-911 caller location identification, and so forth. An electronic museum guide pushes the digital content related to the exhibit that the user is viewing, suggests points of interest, and provides the information the user requests.

The essential techniques of indoor LBSs include indoor positioning and indoor MODB techniques. Some early studies on indoor positioning include Harter and Hopper (1997), Priyanthat, Chakraborty, and Balakrishnan (2000) and Want, Hopper, Falcao, and Gibbons (1992). Bahl and Padmanabhan (2000) presented an indoor positioning technique that applied the K-NN (K Nearest Neighbors) method to UDP (User Datagram Protocol) signal strengths. However, these early studies require special equipment dedicated to positioning. Indoor positioning techniques based on a WLAN (Wireless Local Area Network) do not require such dedicated equipment (Ito & Kawaguchi, 2005; Lassabe, Canalda, Chatonnay, & Spies, 2005; Lin & Lin, 2005). To improve the efficiency of WLAN-based indoor positioning, Yim, Park, Joo, and Jung (2008), Yim, Ko, Do, Joo, and Jeong (2008) and Yim (2008) introduced a Kalman filter method and a decision tree method.

MODB updating strategies represent one of the key factors influencing the efficiency of the MODB. If the frequency of updating is high, the communication and process costs are also high. On the other hand, if the frequency is too low, the deviation between the user's actual location and recorded location becomes too large. In this regard, this paper proposes a Kalman filter method of updating the MODB; this method reduces the communication cost and maintains the deviation within a certain threshold.

The main idea behind our strategy is the application of the Kalman filter to the positions measured at $t_0, t_1, \ldots, t_i$ to estimate the state of the terminal at t_i and extrapolate the positions at moments $t_{i+1}, t_{i+2}, \ldots$ with the estimated state. The state of a moving object consists of its position, velocity, and acceleration. For a pedestrian, we safely assume that its acceleration is zero. If the difference between the extrapolated position and the measured position is within a given threshold, the strategy omits the transmission of the measured position.

Considering Fig. 1, let the two solid lines represent the pedestrian's actual track and the dots labeled $t_0, t_1, \ldots, t_5$ represent the measured positions at times $t_0, t_1, \ldots, t_5$. If we know the pedestrian's exact position, velocity, and acceleration at t_5, we can correctly predict his or her positions at $t_6, t_7, \ldots, t_k$. The predicted value for time t_{k+1} is the position of the circle labeled t_{k+1} in the figure. On the other hand, the measured position at t_{k+1} can be the dot labeled t_{k+1} in the figure. Similarly, the circles in the figure represent the predicted values, whereas the dots represent the measured positions. As shown in Fig. 1, as time flows, the difference between the predicted value and the measured value becomes greater.

If our updating strategy is applied to the scenario depicted in Fig. 1, it applies the Kalman filter process to the dots labeled $t_0, t_1, \ldots, t_5$ to estimate the moving object's position and velocity at time t_5. With the estimated values, it predicts the moving object's positions and skips updating up to time t_k. Sometime after t_k, it starts to send measurements again.

Fig. 1. The proposed updating strategy.

2. Related work

As we propose a Kalman filter method for updating indoor MODBs, we review here GPS-based outdoor MODB techniques and the Kalman filter.

Some of the important research areas in the MODB field are as follows: models of location information, techniques of uncertainty management, query languages accessing spatiotemporal data, indexing techniques, data mining, and security. Wolfson (2002) has provided a review of these topics. The location information in a MODB is naturally uncertain. Olston and Widom (2000) introduced a method of handling uncertainty, and Tao, Xiao, and Cheng (2007) introduced the "probably restricted rectangle" and an information retrieval method using the rectangle.

The positions of aircrafts are important information in national defense systems. Kilimci and Kalipsiz (2007a, 2007b) introduced a moving object database system for military command. The main elements of the system included a traditional relational DBMS, a spatial wrapper of the DBMS, and an API component between the GUI and the DBMS. Ooi, Huang, Lin, Lu, and Xu (2007) presented an MODB prototype designed to be implemented on top of a traditional relational DBMS. Jin, Wan, and Yue (2008) proposed a CASE tool supporting spatiotemporal conceptual modeling for moving object database applications. The tool has the following two advantages: (1) It supports both spatiotemporal and non-spatiotemporal characteristics and is compatible with the ER model. (2) It supports systematic spatiotemporal characteristics and can represent different spatiotemporal changes for diverse moving object applications. Guting (2007) explained how the SECONDO can be used in building a moving object database system.

The query-based indexing technique involves the indexing of queries regarding moving objects, not the indexing of locations of the objects. The location-based indexing technique can be divided into two categories: indexing current and future positions and indexing past positions. These indexing methods are all variants of R-Tree. Makki, Sun, and Khojastehpour (2009) introduced the Delineated R-Tree (DR-Tree) indexing structure, which is highly balanced and has performance advantages over other R-tree-based indexing methods. Han et al. (2005) proposed another new indexing method, the S-TB Tree method. Zhang, Li, and Zhang (2006) proposed an indexing method of efficiently processing historical spatiotemporal pattern queries.

Currently, there is no commercially available moving object database. MODB applications inevitably provide a map on their user interface. Kilimci and Kalipsiz (2007a, 2007b) introduced a digital map to indicate the locations of static or moving objects. They used Oracle's Spatial SDOAPI Java Class Library to handle spatial data. Kuijpers and Vaisman (2007) presented a formal model in which the geometric components of the thematic layers in a GIS are represented as an OLAP (On Line Analytical Processing) dimension hierarchy and introduced the notion of spatial aggregation. They then addressed the moving object aggregation over a GIS.

Advances in Global Positioning Systems, wireless communication systems, and the miniaturization of computing devices have resulted in the emergence of various applications for location-based services (LBSs). As a result, there is an increasing need for the efficient management of vast amounts of location-in-time information for moving objects. Therefore, one of the most important issues the MODB must address is the spatio-temporal language. The queries that a MODB should be able to handle include the determination of the similar trajectories of a moving object. With this kind of query, we can determine the migratory patterns of animals or recommend a route to a driver. A Nearest Neighbor (NN) query is defined as follows: given a set of trajectories T and a trajectory Q, find the trajectory in T that is closest to Q. If we find

two closest trajectories instead of just one, then the query is called 2NN. Xiao (2009) addressed a Set Nearest Neighbor query. An example KNN (K-Nearest Neighbor) query is "What will be the query's 3NN 2 minutes from now?" Liao, Wu, Zhang, and Zhong (2009) introduced a new multi-threaded and cache-conscious algorithm to efficiently process massive, continuous KNN queries. Iijima and Ishikawa (2009) proposed techniques including pruning candidate objects for processing a Nearest Neighbor query when the location of the query object is specified by an imprecise Gaussian distribution. Ding, Trajcevski, and Scheuermann (2008) proposed a Frechet distance-based approach to the similarity join for large sets of moving object trajectories and illustrated that the proposed method was 50% faster than the traditional methods.

There are two types of moving objects: moving points such as vehicles and moving regions such as hurricanes and oil spills. Spatial objects evolve as time flows, and their topological relationships develop over time. Spatio-temporal predicates have been proposed to examine these time-varying relationships. Schneider (2005) proposed a generic algorithmic scheme for evaluating spatio-temporal predicates. The following is an example of a continuous spatio-temporal query: "Continuously inform commander C all friendly units within 10 miles from soldier S." To minimize the execution cost of such queries, Xiong, Elmongui, Chai, and Aref (2007) proposed the Query–Track–Participate (QTP) query processing model, where a query is continuously answered by a querying server, a tracking server, and a set of participating servers. Frihida, Zheni, Ghezala, and Claramunt (2009) introduced an algebraic Spatio-Temporal Trajectory data type (STT) for the representation of the object trajectory. The STT is an abstract data type endowed with a set of operations designed as a way to cover the syntax and semantics of a given trajectory.

Wolfson, Sistla, Chamberlain, and Yesha (1999) thoroughly discussed updating policies for the MODB. They introduced three updating policies for the MODB: speed dead-reckoning (sdr), adaptive dead reckoning (adr), and disconnection-detecting dead-reckoning (dtdr). A dead-reckoning update policy updates the moving object's database location only when the difference between its actual location and its database location is greater than the threshold th. The threshold th is a constant during the trip in the case of sdr, whereas it is newly computed whenever the database location is updated in the case of adr. In the case of dtdr, the threshold is not only newly computed whenever the database location is updated, but it also decreases as the time interval since the last update increases.

The typical applications of the MODB include intelligent transport systems, digital battlefield, location e-commerce, and so forth. A query that must be evaluated for a time interval (e.g., "List the nearest gas stations to my vehicle in the next 15 min") is called a continuous query. Existing MODBs work well for instantaneous queries but not for continuous ones. Li, Huang, and Bian (2009) proposed a new updating strategy that improves the quality of continuous query results and provided their simulation results to demonstrate the effectiveness of their updating strategy. In their simulation, they used a standardized random number generator to assign velocity to the moving object. This implies that they assumed that a moving object was equipped with a device (e.g., a GPS) that could provide the velocity of the moving object. In indoor positioning, there is no device that can provide the velocity of the moving object. In this regard, we propose a Kalman filter method as the velocity predictor.

The main contribution of the present paper is a Kalman filter method that can be used to estimate a moving object's position and velocity. The estimated values can be used to predict the moving object's future positions. For future predictions, most techniques use some mathematical formulas of motion derived from recent movements. However, an object's movements are not simple. In this regard, recent studies have used objects' trajectory patterns for predicting their future positions. Jeung, Liu, Shen, and Zhou (2008) introduced a hybrid prediction model in which both mathematical formulas and trajectory patterns are considered. Wang, Wang, Wang, Lv, and Zhang (2006) proposed an asynchronous position updating algorithm for continuous queries regarding moving objects; the algorithm was found to improve accuracy, reduce the communication cost, and balance the server load more evenly. An example of a continuous query is "All vehicles that appear in the region R in the next 10 min."

Processing the location stream, which is too fast for the MODB can handle, is one of the problems that we have to solve to build a high-performance MODB. Yu, Chen, Rao, and Liu (2004) proposed a location filter to address the problem of processing the location stream. The location filter consists of an online location filter and a temporal location manager. The online location filter executes the filtering algorithm plug-ins and filters the location stream online while the temporal location manager inserts location data into the database at an acceptable insertion rate.

Our strategy is a type of sdr. However, the sdr proposed by Wolfson et al. (1999) cannot be directly used for an indoor MODB because it assumes that a moving object knows its velocity and is a vehicle equipped with a GPS (Global Positioning System) unit. The GPS provides a moving object's location and velocity. Making use of the velocity, the updating policies predict the moving object's future locations, and the predicted locations are used as the object's database locations. The main idea behind this paper is to use the Kalman filter to estimate the indoor moving object's velocity.

The Kalman filter is a recursive process of estimating a state when the state is governed by the following equation:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}, \tag{1}$$

with a measurement

$$z_k = Hx_k + v_k, \tag{2}$$

where the random variable w represents the process noise, v represents the measurement noise, and the probability distribution functions are

$$p(w) \sim N(0, Q)$$

and

$$p(v) \sim N(0, R).$$

The Kalman filter process is as shown in Fig. 2 (Welch & Bishop, 2006). The process repeats the time update and the measurement update until the termination condition is met. The state of a moving object consists of its location and speed. Therefore, $\hat{x}$ is a vector of 4×1 as shown in Eq. (3), where k stands for the kth iteration, x and y represent the x and y coordinates of the location of the moving object, respectively, and v_{xk} and v_{yk} represent the speed of the moving object along the x-axis and y-axis, respectively

$$\hat{x}_k = \begin{bmatrix} x_k \\ y_k \\ v_{xk} \\ v_{yk} \end{bmatrix}. \tag{3}$$

As the role of the matrix A is to project the state at Δt time units ahead, it is defined as Eq. (4), where Δt is set to 1 for the Kalman filter process. We do not consider any optional control input and omit the Bu_{k-1} term

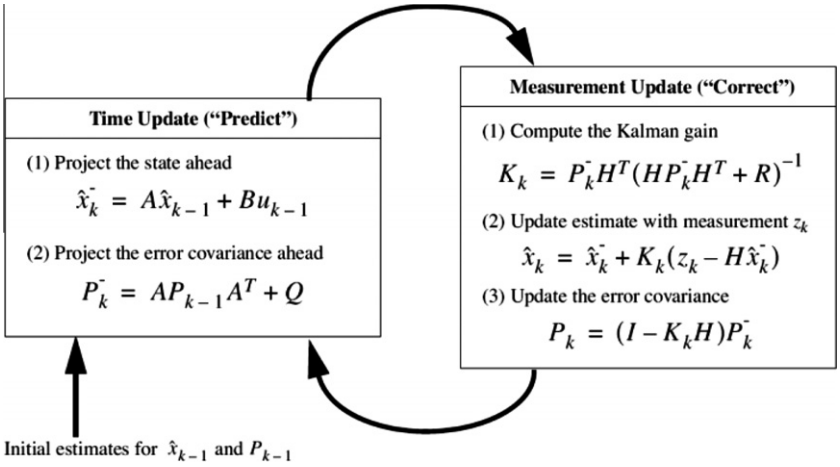

Fig. 2. The Kalman filter process.

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (4)$$

During the time update stage, not only x but the error covariance P is also projected. In the process of projecting P, the *process noise covariance* Q is taken into account.

In the measurement update stage, we adjust $\hat{x}$ with the kth measurement z_k. In this application of the MODB, the position of the moving object is measured. Therefore, z_k is a 2×1 vector consisting of x and y coordinates. As the state of a moving object, $\hat{x}$, is a 4×1 vector, we need a matrix to extract its location, the x and y coordinates, from $\hat{x}$. The matrix is named H and defined as shown in Eq. (5). The matrix R is a 2×2 matrix representing the measurement noise covariance

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \quad (5)$$

3. Kalman filter-based updating strategy

The major components of an indoor MODB system are the database server, the web server, and the communication server. The communication server continuously receives (Time, Location) tuples from mobile terminals. The Location is the coordinates of the measured location of the mobile terminal at the moment of Time. The communication server passes the received tuples to the database server. The web server delivers the user's commands to the database server and displays the answers from the database on the user's terminal. The database server stores all the (Time, Location) tuples from all the mobile terminals in the application field of the MODB. As a mobile terminal sends its location periodically and there are a great number of mobile terminals, the communication cost of MODB updating is very high, and the size of the information stored in the MODB is huge.

In this regard, the present paper proposes an updating strategy with which we can reduce the communication cost and storage without degrading the accuracy of database locations. Our strategy applies the Kalman filter to the positions measured at $t_0, t_1, \ldots, t_i$ to estimate the state of the moving object and extrapolates the positions at moments $t_{i+1}, t_{i+2}, \ldots$ by using the estimated state. If the difference between the extrapolated position and the measured position is not greater than the given threshold, the strategy does not send the measured position. Our strategy performs the extrapolation when the elements of P are all less than 0.001.

The algorithm of our strategy is shown in Table 1. For the first step, the algorithm applies the Kalman filter process shown in Fig. 2 to i consecutively and recently measured positions. We use the K-NN method (Yim, 2008) for the measurement. The formats of the matrices used in Fig. 2 are as follows:

$$X_k = \begin{bmatrix} x_k \\ y_k \\ v_{xk} \\ v_{yk} \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (6)$$

The initial values for the other matrices are determined as follows.

$$Q = \begin{bmatrix} 0.001 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0.001 \end{bmatrix}, \quad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$
$$P_0 = \begin{bmatrix} 300 & 0 & 0 & 0 \\ 0 & 300 & 0 & 0 \\ 0 & 0 & 300 & 0 \\ 0 & 0 & 0 & 300 \end{bmatrix}. \quad (7)$$

After the first step, the algorithm checks the elements of P, and if they are all less than the small constant "Epsilon" (e.g., 0.001), the algorithm performs the extrapolation. In the process of the extrapolation, we replace all the Δts of A with 1s. Then we multiply the A by the ith state obtained in the first step. The result of the multiplication process is assigned to the variable "prediction" in the algorithm. At this moment, if the difference between the $i+1$th measured location and the predicted location is less than the threshold (e.g., 3 m), we do not send the measured location to the communication server. Otherwise, we send the measurement.

If we did not send the $i+1$th measured location, we keep applying the updating algorithm to the $i+2$th measurement, $i+3$th measurement, and so forth. When we apply the algorithm to the $i+2$th measurement, the Δts of A are all set to 2. When we apply the algorithm to the $i+3$th measurement, the Δts of A are all set to 3, and so forth. However, the value of i does not change. In other words, we always apply the Kalman filter to the i recent measurements.

Sometime later, if (|prediction − measurement| < threshold) is no longer true for n (e.g., $n = 3$) consecutive measurements, we can conclude that our prediction is no longer valid and let the mobile terminal send the n measurements to the server.

Table 1
Our algorithm for updating.

```
(1)  Apply the Kalman filter process shown in Fig. 1 to the i measured
     locations to produce x̂_i.
(2)  measurement = i + 1th measurement obtained by our positioning
     algorithm;
(3)  if (elements of P < Epsilon) then {
(4)    prediction = Ax̂_i;
(5)    if (|prediction − measurement| < threshold),
(6)      then do not send the measurement.
(7)      else send the measurement to the server.
(8)  else send the measurement to the server:}
```

When the database server receives a query "Where was the person A at the time xx?" it looks for (xx, Location) in the database. If it finds one, it returns the Location; otherwise, it performs our Kalman filter-based extrapolation and returns the result as the answer to the query. As our updating policy skips updating when the difference between the prediction and the measurement is less than the threshold, the answers returned by the database server are always within the threshold from the measured location.

4. Implementation

We implemented our updating strategy on a laptop computer in C# with Microsoft Visual .NET 2005. The major components of the system were the map rendering, the RSSI scanner, the positioning, and the updating modules. Our implementation of these modules is now described.

4.1. The map rendering module

The purpose of the MODB is to answer spatio-temporal queries. Therefore, most answers are displayed on a map or a drawing for indoor applications. Our map rendering module displays a drawing recorded in an AutoCAD DXF (Drawing Interchange Format or Drawing Exchange Format) file and provides map manipulation functions such as zoom-in, zoom-out, and move. A DXF file consists of a large number of entities. The entities are classified into the arc, line, circle, and polyline types, among others. In DXF, an arc is determined by its attributes such as its center, radius, start angle, and end angle. The attributes of a line are the start and end points. Similarly, the attributes of each entity type is predefined in DXF. In DXF, every entity representation follows the entity type as shown in Table 2. The type of the entity shown in Table 2 is LINE, and the type name LINE follows "0," which indicates that the entity type appears in the next line. The "10" in the table indicates that the X coordinate of the start point appears in the next line. The others are obvious.

Table 2
Selected lines of the DXF file used in the proposed rendering module.

Code	Description
...	
0	Indicates the start of an entity
LINE	Entity type
...	
10	The next line is the X coordinate of the start point
35.88	X coordinate
20	The next line is the Y coordinate of the start point
13.65	Y coordinate
...	Z is omitted
11	The next line is the X coordinate of the end point
58.15	X coordinate
21	The next line is the Y coordinate of the end point
40.68	Y coordinate
...	A lot of entities and others

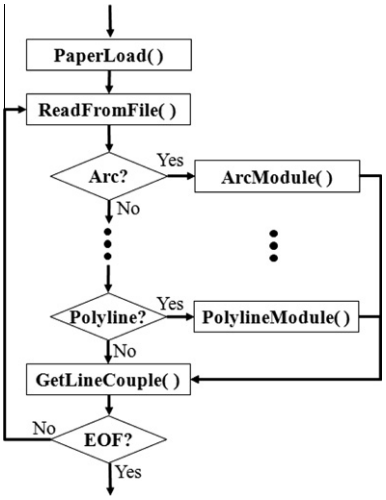

Fig. 3. A flowchart of the input part of our rendering module.

The main parts of our rendering module are the input, drawing, and manipulation parts. A flowchart of the input part is shown in Fig. 3. The first method is PaperLoad( ), which uses the dialog tool of C# and returns the full path of the DXF file. It then continues reading two lines from the DXF file until the end of the file. We read two lines at a time because the entity type follows "0" in the DXF. The entity type determines the way of reading the attributes of the entity. Therefore, if the entity type (for example LINE) is read, we invoke the method (LineModule in this case) which reads in the attributes of the type. For each entity type, we define a class with local variables corresponding to the attributes of the entity. It also has methods including draw( ). For example, x_start, y_start, x_end, and y_end are some of the variables of the LINE class. For each entity read, our input module creates an object and appends it to the list called EntityList.

Our drawing part draws all the objects listed in the EntityList. Before the actual drawing, it calculates the coordinates of the center of the drawing, paperMid and the center of the window, windowMid. The center of the drawing is defined by the following:

$$\begin{aligned} \text{paperMid}X &= (X\text{Min} + X\text{Max})/2, \\ \text{paperMid}Y &= (Y\text{Min} + Y\text{Max})/2, \end{aligned} \tag{8}$$

where XMin is the minimum of x coordinates appearing in the entire DXF file. XMax, YMin, and YMax are all defined similarly. The center of the window is defined by the following:

$$\begin{aligned} \text{windowMid}X &= \text{drawPanelWidth}/2, \\ \text{windowMid}Y &= \text{drawPanelHeight}/2, \end{aligned} \tag{9}$$

where drawPanelWidth (Height) is the width (height) of the panel where the drawing is drawn. We now calculate the scale for the drawing, which is basically the ratio of the window size to the drawing size. It is called mainScale and can be calculated by Eq. (10). We choose the minimum out of scale$X$ and scale$Y$ and then multiply it by 0.9 so that the drawing can easily fit into the window

$$\begin{aligned} &\text{mainScale} = \text{MIN}(\text{scale}X, \text{scale}Y) * 0.9, \quad \text{where} \\ &\text{scale}X = \text{drawPanelWidth}/(\text{Max}X - \text{Min}X), \\ &\text{scale}Y = \text{drawPanelHeight}/(\text{Max}Y - \text{Min}Y). \end{aligned} \tag{10}$$

Given a point (paper$X$, paper$Y$) from the DXF file, we can calculate the point on the window, (win$X$, win$Y$), by using Eq. (11). A point (paper$X$, paper$Y$) in the DXF file is represented at (win$X$, -

win$Y$) on the window. In Eq. (11), we use a single operand negation operation because Y coordinates increase from the top to the bottom for a window, whereas they increase from the bottom to the top for a drawing. Once we have the window coordinates, drawing an entity is just a matter of calling the right method provided by C#

$$\begin{aligned} \text{win}X &= (\text{paper}X - \text{paperMid}X) * \text{mainScale} + \text{windowMid}X, \\ \text{win}Y &= -(\text{paper}Y - \text{paperMid}Y) * \text{mainScale} + \text{windowMid}Y. \end{aligned} \tag{11}$$

The manipulation part of the rendering module provides zoom in, zoom out, up, down, left, right, and so forth. It is easy to implement these functions. For example, zoom in (out) can be done by increasing (decreasing) the value of mainScale. For returning to the original scale, the OriginalScale( ) method is provided. Move up (down) can be done by increasing (decreasing) paperMid$Y$, and move right (left) can be done by increasing (decreasing) paperMid$X$. Converting DXF coordinates into window coordinates can be done by performing Eq. (11). The inverse of Eq. (11) can be used for converting window coordinates into DXF coordinates.

4.2. RSSI scanner

The 802.11 NIC (Network Interface Card) is a wireless LAN card, which is an OSI layer 1 (physical layer) and 2 (data link layer) device because it allows physical access to a networking medium and provides a low-level addressing system through the use of a MAC address. A computer obtains the information (e.g., SSID, BSSID, RSSI, Network type, and so forth) broadcasted by APs through an 802.11 NIC, selects the AP with the strongest RSSI, and communicates with it.

In order for an application program to communicate with a WLAN card, it needs to use the Network Driver Interface Specification (NDIS), which is an application programming interface (API) for network interface cards (NICs). That is, a physical layer device (e.g., 802.3 NIC, 802.5 NIC, 802.11 NIC, etc.) and an application program can communicate with each other through the NDIS. In other words, an application program can read information from an 802.11 NIC by implementing the NDIS IOCTL Interface, which can access the NDIS driver. The NDISUIO driver is a public code that comes as a part of Windows-XP and provides an application programming interface for the NDIS driver.

Windows XP selects a wireless network as the base wireless utilities embedded in Windows XP by using the Wireless Zero Configuration. The Wireless Zero Config Service (WZCSVC) provides the initializing connection service, the monitoring device status, and the driver communication with the OID designating 820.11. As Windows XP automatically uses the WZCSVC, we need to disable Windows XP's Wireless LAN configuration to use the WZCSVC.

An application program can communicate with the 820.11 NIC through the NDISUIO driver by using the WZCSVC service. The process of communication between an application program and a wireless LAN card is summarized in Table 3. Yim, Park, et al. (2008) and Yim, Ko, et al. (2008) described the details of the RSSI scanner implementation.

4.3. The Kalman filter

Our Kalman filter-based updating strategy estimates the mobile terminal's state by applying the Kalman Filter process to i consecutively measured locations. Our algorithm for the Kalman filter process is shown in Table 4. The first step initializes X with the first measured location and P with the matrix shown in (Eq. (7)). The variables – X_matrix, P_matrix H_matrix, K_matrix, I_matrix, R_matrix, Q_matrix, and A_matrix – are the objects of the Matrix class defined in the algorithm and represent the variables X, P, H, K, I, R, Q, and A in Fig. 2, respectively. Using the Matrix class, a vector can be represented by specifying the size of the second dimension as 1. Therefore, the vector X is represented by X_matrix in the algorithm. The Matrix class has +, −, * operators for matrix plus, minus, and multiplication. The multiplication operator is overloaded as it can also be a scalar multiplication if the parameters are a scalar and a matrix. Some methods for matrix manipulations, including transposing, copying, and inversing matrix, are also defined in the Matrix class.

Table 3
A summary of the process of communication between an application program and a wireless LAN card.

Disable Windows XP's WZCSVC
② Access the NDISUIO driver by using the CreateFile( ) function in kernel32.dll. CreateFile( ) returns the handle
③ Communicate with the device via the DeviceIoControl( ) function in kernel32.dll
④ Release the device by using CloseHandle( ) when the communication is over and enable the WZCSVC

Lines 2 and 3 perform the "time update" and lines 4–7 compute the "Kalman gain." During the computation of the Kalman gain, the algorithm determines whether Temporary_K is invertable. If it is not, the algorithm aborts the process. Line 8 updates the estimate with ith measurement z_i, and line 9 updates the error covariance.

4.4. Positioning method and test

Our positioning method is a WLAN-based K-NN method. This method is a type of fingerprinting method. For an IMODB in a WLAN environment, a fingerprint is a set of RSSIs (Received Signal Strength Indicators) scanned from the wireless LAN card. The deployment of the method consists of an offline phase and an online phase. In the offline phase, we build a lookup table with the training data (fingerprints). An example training data set is shown in Table 5. In the table, AP and CP stand for the access point and the candidate point, respectively. The entire area is covered by a rectangular grid of points called *candidate points*. At each of the candidate points, we scan the RSSIs many times. Let RSSI_{ij} denote the jth received RSSI sent by AP_i. A row of the lookup table is an ordered pair of (coordinate, a list of RSSIs). A coordinate is an ordered pair of integers (x, y) representing the coordinates of a candidate point. A list of signal strengths consists of five integers, $\text{RSSI}_1, \text{RSSI}_2, \ldots$, where RSSI_i is an average of RSSI_{ij} received at (x, y) and sent by AP_i. A sample lookup table is shown in Table 5.

In the online phase, the positioning program gathers the RSSIs the user receives at the moment. If the positioning program is running on the user's handheld terminal, the terminal itself collects the RSSIs. Let $X = (-40, -56, -54, -69, -66)$ be the vector of the collected RSSIs. The K-NN then compares X with each row of the lookup table and finds the closest candidate point, returning it as the user's current location. If K equals 2, it finds the two closest candidate points and returns the average of the two points as the user's current location.

Our MODB was installed on a workstation. For the test, we collected the measured locations by executing the indoor positioning module once every second while we walked along a corridor shown as the thick solid lines in Fig. 4. The speed of our walk was 0.5 m/s, and the number of measurements was 142 per walk. A typical result of positioning is shown in Fig. 5.

We walked the corridor 25 times and applied our updating method to the 142 measurements of each walk. Fig. 6 shows the results of the first walk, which used 0.05 and 3 m for the values of Epsilon and threshold, respectively. In this experiment,

Table 4
The proposed Kalman filter algorithm.

```
// step (1) Initial guess
1    initializes X_matrix with z1 and P_matrix as shown in (Eq. (8));
// apply the Kalman Filter process to i measured locations
     For (j = 0; j < i; j++) {
// step (2) Time Update
2    X_matrix = A_matrix * X_matrix;
3    P_matrix = (A_matrix − P_matrix * A_matrix.Transpose) + Q_matrix;
// Kalman Gain
4    Temporary_K = H_matrix * P_matrix * H_Matrix.Transpose( ) + R_matrix;
5    Inverse_K = Temporary_K.Inverse( );
// If K is not invertable, terminate
6    if(Inverse_K is not the inverse of K), return false;
7    K_matrix = P_matrix * H_matrix.Transpose( ) * Inverse_K;
// Update estimates with the measurement zi
8    X_matrix = X_matrix + K_matrix(zi − H_matrix * X_matrix);
// Update the error covariance
9    P_matrix = (I_matrix − K_matrix * H_matrix) * P_matrix;
     }
```

Table 5
A sample lookup table of the K-NN (C.P stands for candidate points, CP*i* is the coordinates of *i*th C.P, AP*i* is the MAC address of the *i*th AP).

C.P	AP				
	AP1	AP2	AP3	AP4	AP5
CP1	−39	−55	−56	−70	−67
CP2	−40	−56	−55	−69	−66
CP3	−44	−42	−62	−45	−61
...	...	...	...	...	...

"elements of $P <$ Epsilon" were met at the 12th measurement for the first time. At that time, the value of $\hat{x}_i$ was as follows:

$$\hat{x}_i = \begin{bmatrix} 18788.62 \\ 40568.56 \\ 434.53 \\ -138.75 \end{bmatrix}.$$

The values 18788.62 and 40568.56 of $\hat{x}_i$ represent the coordinates, and 434.53 and −138.75 represent the velocity of the mobile terminal. The dots in the figure represent the measurements that were transmitted.

Our experimental results are summarized in Table 5. When we set the threshold to 3 m, our strategy skipped sending measurements 74.6 times on average. This accounted for 52.5% of the observed measurements. As shown in Table 6, the average error of our measured positions was 2.924, which was very close to our threshold. This might have been the reason why the skip rate was 52.5%. In addition, the average error (3.05214) of the recorded positions of our policy was slightly higher than that (2.924) of "without updating policy," but they were very close. This was expected because we started sending measurements when "(|prediction − measurement| < threshold)" held.

In another set of experiments, we set the threshold to 2 m. Our strategy skipped sending only 37 times (i.e., 26%) on average. This was expected because a lower threshold value means less skips. As the number of skips decreased, the average error of the recorded measurements of this experiment neared that of "without updating policy."

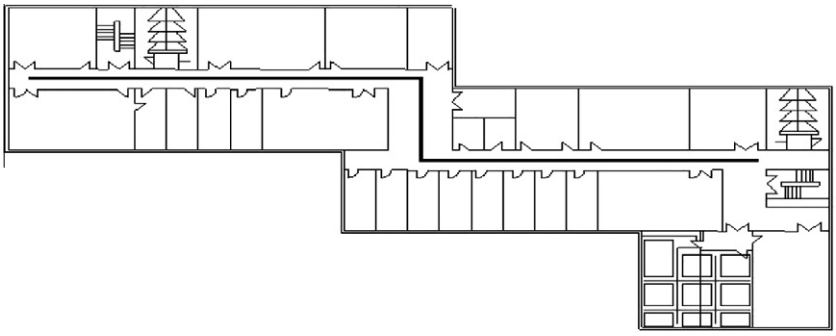

Fig. 4. Our test bed.

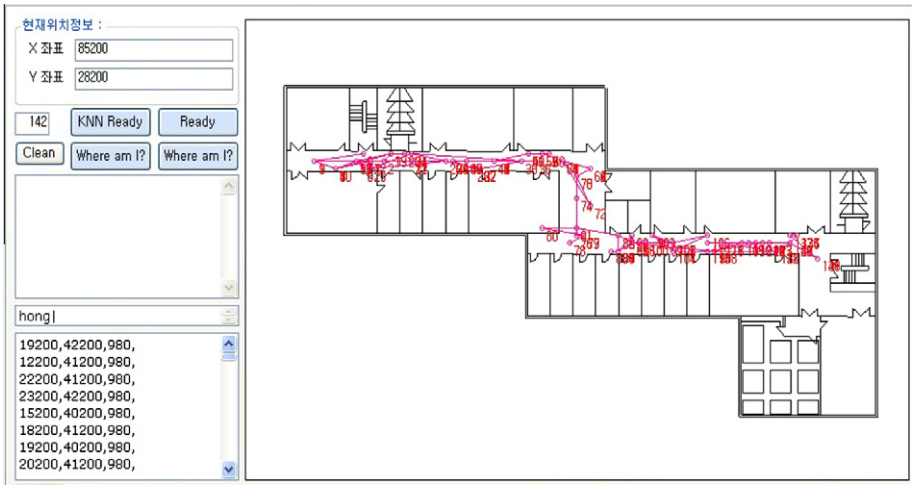

Fig. 5. A typical result of positioning.

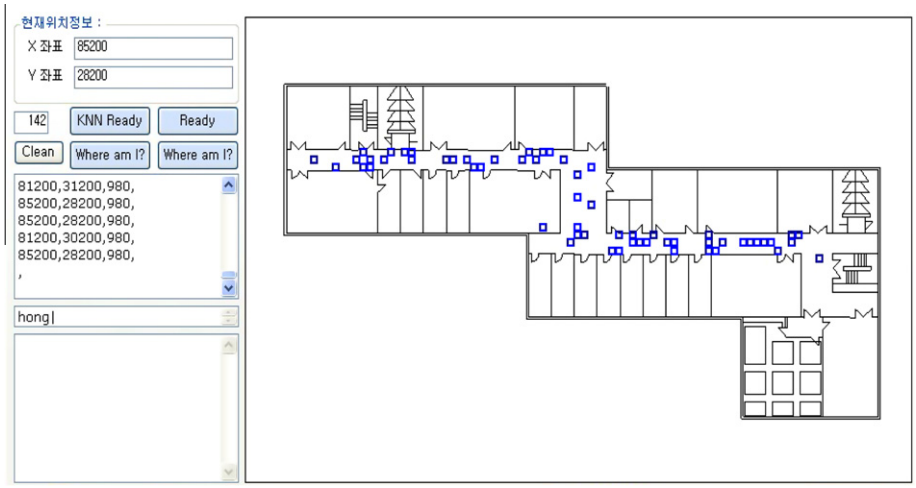

Fig. 6. The dots represent the measurements transmitted.

Table 6
Summary of experiments.

	Without updating policy	Threshold = 3 m	Threshold = 2 m
Number of skips	0	74.6 (52.5%)	37 (26%)
Average error (m)	2.924	3.05214	2.956

These results indicate that our updating method can reduce the communication cost by approximately half and that the reduction entails almost no trade-off for the indoor moving object database when the position of moving objects is measured by the WLAN-based K-NN positioning method.

We expected that our updating algorithm would reduce the communication cost by more than 70% in the test bed because the track consisted of only three straight lines. A 52.5% reduction was not satisfactory. This disappointing test result stems from an assumption of the Kalman filter. The Kalman filter assumes that measurements are normal distributed around the real value. This implies that if we perform our positioning algorithm many (100) times at a point P without moving around, the average of the measured positions should be P. In reality, the average of the positions measured by our positioning algorithm did not exactly coincide with the real position.

5. Conclusion

One of the most important research topics in the moving object database field is the identification of updating strategies that can reduce communication costs and storage space. This paper introduced a Kalman filter updating method for indoor moving object databases and implemented the prototype updating system. Our updating strategy estimates an indoor mobile object's velocity by using the Kalman filter and predicts the object's position by using the estimated velocity.

We implemented many modules, including map rendering module, RSSI scanner, positioning module, database server, client module, communication module, and so on. Among them, the map rendering module, RSSI scanner, the Kalman filter and positioning module were described in detail in this paper.

To verify the effectiveness of our updating strategy, we performed a number of experiments. The results indicate that our updating method can reduce the communication cost by approximately half and that the reduction entails almost no trade-off for the indoor moving object database when the position of moving objects is measured by the WLAN-based K-NN positioning method.

The 50% reduction is not satisfactory. This disappointing performance was due to the assumption of the Gaussian distribution the Kalman filter makes. In this regard, future research is warranted to develop a better prediction method by identifying the track patterns. At the same time, we are planning to develop the system such as a smart museum guide enabling practical indoor location-based services in real world.

References

Bahl, P., & Padmanabhan, V. (2000). RADAR: An in-building RF-based user location and tracking system. In *Proceedings of INFOCOM* (pp. 775–784).

Ding, H., Trajcevski, G., & Scheuermann, P. (2008). Efficient similarity join of large sets of moving object trajectories. In *Proceedings of 15th international symposium on temporal representation and reasoning* (pp. 79–87).

Frihida, A., Zheni, D., Ghezala, H., & Claramunt, C. (2009). Modeling trajectories: A spatio-temporal data type approach. In *Proceedings of 20th international workshop on database and expert systems application* (pp. 447–451).

Guting, R. (2007). How to build your own moving objects database system. In *Proceedings of international conference on mobile data management* (pp. 1–2).

Han, L., Wu, J, Xie, K., Ma, X., Xu, D., Zhang, H., et al. (2005). A spatio-temporal database prototype for managing moving objects in GIS. In *Proceedings of IEEE international geoscience and remote sensing symposium*.

Harter, A., & Hopper, A. (1997). A new location technique for the active office. *IEEE Personal Communications, 4*(5), 43–47.

Iijima, Y., & Ishikawa, Y. (2009). Finding probabilistic nearest neighbors for query objects with imprecise locations. In *Proceedings of 10th international conference on mobile data management: Systems, services and middleware* (pp. 52–61).

Ito, S., & Kawaguchi, N. (2005). Bayesian based location estimation system using wireless LAN. In *Proceedings of the 3rd IEEE international conference on pervasive computing and communications workshops (PerCom 2005 workshops)* (pp. 273–278).

Jeung, H., Liu, Q., Shen, H., & Zhou, X. (2008). A hybrid prediction model for moving objects. In *Proceedings of IEEE 24th international conference on data engineering* (pp. 70–79).

Jin, P., Wan, S., & Yue, L. (2008). Conceptual modeling for moving objects database applications. In *Proceedings of 9th international conference on mobile data management* (pp. 217–218).

Kilimci, P., & Kalipsiz, O. (2007a). Geometric modeling and visualization of moving objects on a digital map. In *Proceedings of geometric modeling and imaging (GMAI'07)* (pp. 39–43).

Kilimci, P., & Kalipsiz, O. (2007b). Moving objects databases in space applications. In *Third international conference on recent advances in space technologies* (pp. 106–108).

Kuijpers, B., & Vaisman, A. (2007). A data model for moving objects supporting aggregation. In *Proceedings of IEEE 23rd international conference on data engineering workshop* (pp. 546–554).

Lassabe, F., Canalda, P., Chatonnay, P., & Spies, F. (2005). A Friis-based calibrated model for WiFi terminals positioning. In *Proceedings of 6th IEEE international symposium on a world of wireless mobile and multimedia networks (WoWMoM 2005)* (pp. 382–387).

Li, X., Huang, M., & Bian, F. (2009). An adaptive dead-reckoning updating policy for continuous query. In *Proceedings of international conference on artificial intelligence and computational intelligence* (pp. 194–198).

Liao, W., Wu, X., Zhang, Q., & Zhong, Z. (2009). Improving throughout of continuous k-nearest neighbor queries with multi-threaded techniques. In *Proceedings of IEEE international conference on intelligent computing and intelligent systems* (pp. 438–442).

Lin, T. N., & Lin, P. C. (2005). Performance comparison of indoor positioning techniques based on location fingerprinting in wireless networks. In *Proceedings of the 2005 international conference on wireless networks, communications and mobile computing* (Vol. 2, pp. 1569–1574).

Makki, S., Sun, B., & Khojastehpour, M. (2009). An efficient information access scheme for mobile objects. In *Proceedings of IEEE international conference on information reuse and integration* (pp. 312–317).

Olston, C., & Widom, J. (2000). Offering a precision-performance tradeoff for aggregation queries over replicated data. In *Twenty-sixth international conference on very large data bases, Cario, Egypt*.

Ooi, B., Huang, Z., Lin, D., Lu, H., & Xu, L. (2007). Adapting relational database Engine to accommodate moving objects in SpADE. In *Proceedings of IEEE 23rd international conference on data engineering* (pp. 1505–1506).

Priyanthat, N., Chakraborty, A., & Balakrishnan, H. (2000). The Cricket location-support system. In *Proceedings of 6th ACM international conference on mobile computing and networking, Boston, MA*.

Schneider, M. (2005). Evaluation of spatio-temporal predicates on moving objects. In *Proceedings of 21st international conference on data engineering* (pp. 516–517).

Tao, Y., Xiao, X., & Cheng, R. (2007). Range search on multidimensional uncertainty data. *ACM Transactions on Database Systems, 32*(3), 1–54.

Wang, X., Wang, S., Wang, Z., Lv, T., & Zhang, X. (2006). A new position updating algorithm for moving objects. In *Proceedings of 1st international multi-symposiums on computer and computational sciences* (pp. 496–503).

Want, R., Hopper, A., Falcao, V., & Gibbons, J. (1992). The active badge location system. *ACM Transactions on Information Systems, 10*(1), 91–102.

Welch, G., & Bishop, G. (2006). *An introduction to the Kalman filter*. Available at. <http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html>.

Wolfson, O. (2002). Moving objects information management: The database challenge. In *Proceedings of the 5th workshop on next generation information technologies and systems (NGITS'2002), Dan Caesarea Hotel, Caesarea, Israel* (pp. 1–13).

Wolfson, O., Sistla, A. P., Chamberlain, S., & Yesha, Y. (1999). Updating and querying databases that track mobile units. *Distributed and Parallel Databases Journal (DAPD) on Mobile Data Management and Applications, 7*(3), 257–288.

Xiao, Y. (2009). Set nearest neighbor query for trajectory of moving objects. In *Proceedings of 6th international conference on fuzzy systems and knowledge discovery* (pp. 211–214).

Xiong, X., Elmongui, H., Chai, X., & Aref, W. (2007). Place: A distributed spatio-temporal data stream management system for moving objects. In *Proceedings of international conference on mobile data management* (pp. 44–51).

Yim, J. (2008). Introducing a decision tree-based indoor positioning technique. *Expert Systems with Applications, 34*(2), 1296–1302.

Yim, J., Ko, I., Do, J., Joo, J., & Jeong, S. (2008). Implementation of a prototype positioning system for LBS on U-campus. *Journal of Universal Computer Science, 14*(14), 2381–2399.

Yim, J., Park, C., Joo, J., & Jung, S. (2008). Extended Kalman Filter for wireless LAN based indoor positioning. *Decision Support Systems, 45*, 960–971.

Yu, X., Chen, Y., Rao, F., & Liu, D. (2004). Filtering location stream in moving object database. In *Proceedings of 15th international workshop on database and expert systems applications* (pp. 645–649).

Zhang, W., Li, J., & Zhang, W. (2006). Spatio-temporal pattern query processing based on effective trajectory splitting models in moving object database. In *Proceedings of 1st international multi-symposiums on computer and computational sciences* (pp. 540–547).