# An Outdoor Recommendation System based on User Location History

Yuichiro Takeuchi and Masanori Sugimoto

***Abstract*—Recommendation systems are now widely used in online shopping sites, but so far there have been few attempts of applying them to real-world shopping. In this paper we propose a novel real-world recommendation system, which recommends shops to users based on their individual preferences and needs, estimated by analyzing their past location history acquired using GPS. The system automatically figures out each user's frequently visited shops using a custom estimation algorithm, and makes recommendations by using the shops as input to the item-based collaborative filtering algorithm. Furthermore, to provide more timely recommendations, our system takes into account the user's usual shopping routes, and the ease of access from the user's current location to each shop. We have conducted an evaluation test using a prototype of our system at Daikanyama, Tokyo, and the results show great promise in the system's ability to make accurate recommendations.**

***Index Terms*—Collaborative Filtering, Location-Aware Systems, Location Discovery, Recommendation Systems**

## I. Introduction

TODAY, recommendation systems are used in many online shopping sites. Their chief function is to recommend items that match each customer's preferences and needs, which are estimated from information gathered through their past activities at the site, for example which items they bought or which items they showed interest in.

Recommendation systems have numerous benefits. Studies in experimental psychology say that, when used effectively, recommendation systems in shopping sites have the ability to increase the perceived credibility of the site[1], thus leading customers into buying more items. It is this ability of generating increased profit that has brought about the current popularity of recommendation systems among shopping sites. When seen from a customer's point of view, recommendation systems are helpful in the sense that they assist them to easily find items that match their tastes. They are perceived as an intelligent tool that effectively helps them out as consumers in this age of information overload.

But despite the benefits that recommendation systems grant to its customers and shopping site owners, we believe that a single fact is severely limiting us from appreciating them to their full potential: the single fact, that recommendation systems can only be used for shopping on the Internet, not for shopping in the city, or in other words, *in the real world*.

Yuichiro Takeuchi and Masanori Sugimoto are with the School of Frontier Sciences, The University of Tokyo, 5-1-5 Kashiwanoha, Kashiwa-shi, Chiba, Japan. (e-mail: takeuchi@itl.k.u-tokyo.ac.jp, sugi@itl.k.u-tokyo.ac.jp).

Fig. 1. Using CityVoyager, a recommendation system for real-world shopping

Despite the growing popularity of online shopping, the majority of shopping activities is still done at real-world shops. Thus it can be inferred that, for us to be able to fully appreciate the benefits of recommendation systems, we must apply them to real-world shopping. The difficulty of this task lies in that it is extremely more difficult to acquire sufficient customer activities needed for estimating preferences in real-world shopping, compared to online shopping where all user activities can be easily recorded in the server.

In this paper, we introduce CityVoyager (Fig. 1), a real-world recommendation system designed to run on mobile devices, which recommends shops to users based on preferences estimated from their past location history. Location data can be easily acquired using means such as GPS, and it contains rich information about each user's personal preferences. Our system effectively applies location data to the widely used item-based collaborative filtering algorithm, by transforming continuously recorded location data into a form of a list that contains each user's {¥it frequently visited shops}, and rating values which indicate how fond the user is of each shop. This list can be directly used as input to the filtering algorithm to make recommendations in the exact same manner as conventional recommendation systems. We have devised a custom algorithm for this transformation of data, which automatically finds each user's frequently visited shops and calculates rating values without any need of explicit user manipulation. We have also enabled the system to take into account information such as the user's usual shopping routes, and the ease of access from the user's current location to each shop, to provide more timely recommendations.

To assess the effectiveness of our system, we have conducted an evaluation test at Daikanyama, one of Tokyo's most revered shopping districts. The results show great promise in the system's ability to make accurate recommendations, although various aspects of the system still needs polish.

## II. Related Work

### A. Location Acquisition

One important characteristic of our system is that it makes extensive use of location data. Numerous studies on location-based systems have been previously conducted, with various methods of location acquisition.

The Olivetti Active Badge[2] and the ParcTab[3] acquire user location using infrared waves. Infrared, being cheap, convenient and unregulated, has long been a popular choice for location acquisition, although it has some disadvantages such as limited accuracy and poor performance in environments abound with obstacles.

RADAR[4][5] detects location using wireless LAN (WLAN). WLAN is increasingly becoming popular as the method of choice for location-aware systems, since in most cases no special equipments are needed, as many recent PCs and mobile devices are already equipped with built-in WLAN capabilities, and the number of hotspots in urban areas is rapidly increasing. RADAR acquires user location from the strengths of signals observed by several WLAN base stations, by using a predefined map consisting of observed signal strengths for sample locations spreading throughout the environment.

The Active Bat[6] system uses ultrasound waves to acquire extremely precise location. The system consists of a small device (a Bat) with ultrasound-emitting capabilities, and a dense array of receivers mounted on ceilings. The relatively slow speed of ultrasound allows the system to correctly measure the distance between each receiver and the Bat, and the precise location of the user can be calculated by triangulation with an accuracy of around 3 centimeters.

The CyberGuide[7] is an example of a system using GPS. The advantages of using GPS are that no equipment other than a GPS receiver is needed, and that fairly high accuracy can be achieved, if only outdoors.

### B. Location Discovery

Our system, CityVoyager, makes recommendations based on information about each user's frequently visited shops, which are automatically estimated by the system. The task of finding frequented places from GPS input (*location discovery*) has been investigated in a number of past researches.

Commotion[8] is one example of a location discovery system. The system tracks users' locations using GPS, and identifies frequently visited locations (buildings), by keeping track of positions where GPS signals were continuously lost. Places where signals were often lost are defined as frequently visited locations. Then, the user can annotate the defined locations with information such as location names, memos, and to-do lists.

Ashbrook and Starner[9], and Zhou et al[10], proposes location discovery methods based on clustering algorithms, which offer more precise results but are more computationally expensive.

The system we propose in this paper finds users' frequently visited buildings in a somewhat similar manner as the above systems, but instead of using a clustering approach, we introduce a completely new algorithm for location discovery, which is much more computationally efficient. This is possible because in our system we only need to search for frequently visited *shops*, which are located in discrete locations, as opposed to conventional location discovery systems which search for frequented *places*, which exist continuously in a two-dimensional space. The discrete nature of shops allows our system to look for frequented places in a much smaller search space, and therefore the computation cost can be reduced greatly.

### C. Recommendation Systems

The concept of recommendation systems have long been explored, and there have already been numerous researches conducted on the field. At the core of recommendation systems is the filtering algorithm, which filters out unnecessary data and decides which data should be recommended to the user. The two most commonly used types of filtering algorithms are *content-based filtering*, and *collaborative filtering*.

*1) Content-based Filtering:* The basic idea of content-based filtering[11][12] is to express the content of each data in a form that can be objectively evaluated, and filter out data whose content doesn't match the user's preferences. The most commonly used method for expressing content is the feature vector method.

According to the feature vector method, the content of each piece of data is expressed in the form of a vector, consisting of values for a set of *features*. Features are defined so that they can effectively convey the content of each data, and that they can be expressed in numerical values. For example, in the case of text data, features are defined as the frequency with which several keywords appear in the text. If the keywords are cleverly chosen, the resulting vector should be able to communicate the content of the text with significant accuracy. The preferences of each user is also expressed as a vector using the same set of features, and if a vector for a piece of data is similar to the vector for the user preference, there should be good chance that the user will like the data, and thus the data is recommended to the user. Most content-based systems are intended only for recommending text data, since appropriately expressing the content of each data is difficult for other types of data.

*2) Collaborative Filtering:* Collaborative filtering[13][14] recommends data that was given high ratings by a number of users, with similar preferences as the user who requested the recommendation.

The first step of collaborative filtering is calculating the similarity of preferences between users. Then, several users with the highest similarity values (*nearest neighbors*) are picked out. Data that has received high ratings among the nearest neighbors, and that the user who requested the recommendation has not yet evaluated, is recommended.

The biggest advantage of collaborative filtering over content-based filtering is that collaborative filtering requires no previous knowledge about the content of the data, and thus can be applied to any type of data, regardless of content.

On the other hand, collaborative filtering also has some disadvantages. First, the only data that can be recommended using collaborative filtering are ones that have already been evaluated by some other user, which means that it may take some time before a piece of data newly introduced in the data space can have a chance of being recommended. When the relative size of the data space is extremely large compared to the number of users, a considerable portion of the data space will not be available for recommendation. Next, collaborative filtering only functions properly when there are users with similar preferences. When the number of users is small, the nearest neighbors found by the system might not necessarily have similar preferences, which may result in the system producing inaccurate recommendations. Finally, the computation cost of collaborative filtering can be a problem, especially when the number of users is large.

There is a variation of collaborative filtering called item-based collaborative filtering[15]. In this approach, instead of calculating the similarity between users, the similarity between items are calculated. Items which show high similarity with the items that the user has given high ratings are recommended.

### D. *Recommendation using Location*

There have been numerous studies of recommendation systems and shopping assistance systems which make use of location data, but so far they seem to be limited to using only the *current* user location. The personal shopping assistant[16] by Asthana et al. presents users with information on special deals according to their current location. Pilgrim[17] is a website recommendation system which takes into account the location from which the user had accessed websites. To the best of our knowledge, our proposed system is the only recommendation system which uses the *history* of continuously recorded location data for recommendation.

## III. System Overview

The basic idea of our recommendation system, CityVoyager, is to estimate the users' individual preferences from the history of their location data collected using GPS, and recommend shops upon request. The system is intended to be useful for various kinds of shoppers, in various situations. For example, the system can be helpful for shoppers new to the area wanting to find shops that match their tastes, or for shoppers more familiar to the area willing to try something new.

Fig. 2 illustrates how CityVoyager compares with conventional recommendation systems for online shopping.

Whereas conventional systems estimate users' preferences from their online activity records, such as items bought or

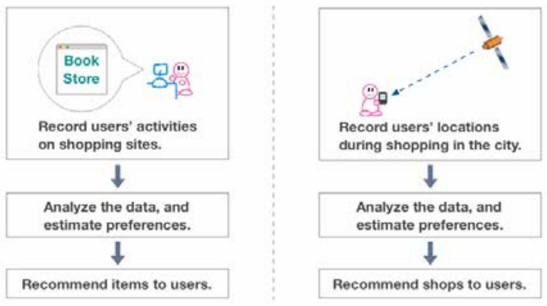

Book Store
Record users' activities on shopping sites.
Analyze the data, and estimate preferences.
Recommend items to users.
Record users' locations during shopping in the city.
Analyze the data, and estimate preferences.
Recommend shops to users.

Fig. 2. Comparison of CityVoyager(right) with conventional recommendation systems(left)

checked in the past, our system estimates preferences based on their location history during shopping in the city.

It should be noted that our recommendation system recommends shops, as opposed to conventional systems in shopping sites which recommends items. We dismissed the idea of recommending items, for the two reasons discussed below.

First, in order to recommend items, information about each specific item that the user has bought or has showed interest in must be obtained, to estimate preferences. In online shopping, these information can be easily acquired from the server log. But in the real-world, these information can only be acquired if every item is equipped with a smart tag like an RFID, or every shop employs a strict customer surveillance system. The latter method is obviously unrealistic, due to privacy concerns. The former method seems more plausible, since smart tags are assumed to become pervasive and replace bar codes in the coming years. However, if we are to base our recommendation system on smart tags, the issue of *coverage rate* must be considered. If every item in every shop becomes equipped with a smart tag, the coverage rate will be 100%. But considering the current coverage rate of bar codes, assuming that the actual rate will become anything close to this is unreasonable. Therefore, a recommendation system based on smart tags will inevitably be a crippled one, since it automatically excludes a considerable portion of items sold in the city. In contrast, our system can include every shop except those in places where GPS does not work, for example inside large building, and the coverage rate is relatively high. The rate should become even higher in the near future, with the advance of alternative location acquisition techniques like the Wi-Fi.

Next, there is no effective filtering algorithm applicable for recommending real-world items. Content-based filtering is almost exclusively used for recommending text data, which is rarely seen in real-world shopping. Collaborative filtering is incapable of dealing with the extremely fast cycle in which new items appear and disappear in the real world, since it cannot recommend items that had not yet been evaluated by a respectable number of users. The algorithm can only work for items like books which are continuously sold for a significant time span, or items that are produced in large quantities. In case of items which are produced in scarce numbers and sold for only a short period of time, for example fashion items made

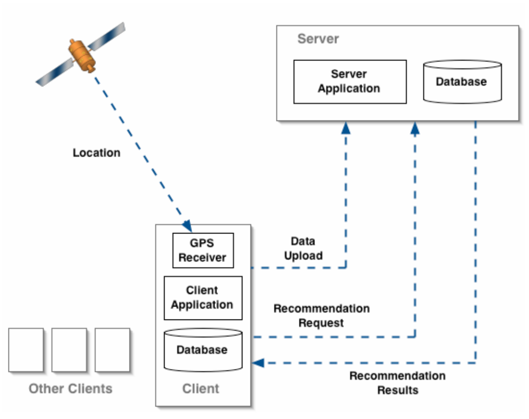


Fig. 3. The CityVoyager system architecture

by lesser known producers, sufficient evaluation results needed for collaborative filtering to properly function cannot be achieved. In contrast, our system is unaffected by this fact because shops exist for a relatively long period of time, enough for gathering evaluations.

CityVoyager is solely based on location data acquired using GPS, and does not require any other sensors. Considering the growing popularity of GPS receivers and GPS-embedded mobile phones, a system that can function using only GPS has a chance of being widely used, and developing such a system should be a worthy attempt.

Fig. 3 illustrates the system architecture of CityVoyager. The main components of our system are client devices carried by users, and a server that performs recommendations. The client device can be any mobile computer device which is or can be equipped with Internet and GPS capabilities. Potential client platforms include PDAs, notebook PCs and mobile phones.

Our system analyzes, and transforms raw location data received from GPS into a list of each user's frequently visited shops. This list is used as input to our filtering algorithm, and so it must be kept inside the server in order to carry out recommendations. If the client device has enough memory space and high computational capabilities, the transformation of location data can be done inside the client. But if the capabilities of the client device is insufficient, the transformation must be done inside the server, and thus the users' raw location data has to be sent to the server. This results in decreased privacy, and should be avoided if possible.

Below, we describe the process by which the system makes recommendations. The recommendation process can be divided into two phases, the data acquisition phase and the recommendation phase.

## A. Data Acquisition Phase

In the data acquisition phase, raw location data from GPS is reconstructed into a list of each user's frequently visited shops. The process can be further divided into three sub-phases: monitoring user location, detecting visits to shops, and finding frequently visited shops.

*1) Monitoring user location:* The location of the user is consistently monitored using GPS. Data acquired by GPS contains errors deriving from a variety of causes. Even in ideal conditions where no tall buildings are present, an error of around 10 meters will inevitably exist, mainly due to the effects of the ionosphere. Also, since GPS signals cannot penetrate through building walls, the system cannot acquire the user's location when he/she is indoors. The location of the user is periodically recorded into a database, inside the client device or the server if the client has insufficient memory. This information is used later to identify the user's usual shopping routes through the city.

*2) Detecting visits to shops:* We exploit the fact that GPS signals cannot penetrate through walls, to detect if the user is indoors or outdoors. But naively using the loss of GPS signals as a proof that the user is inside leads to frequent errors. There are two possible user situations when GPS signals cannot be received: either the user is inside a building, or is surrounded by tall buildings and signals are blocked. On the other hand, simply believing that the user is outdoors because of the availability of GPS signals also leads to errors, since the user may either be outdoors, or inside a building that allows GPS signals to penetrate through its walls due to its structure and building materials (buildings that have high ceilings and walls consisting mainly of glass often fit into this category).

To reduce these errors, we incorporate two timers for indoor and outdoor detection (Fig. 4) The indoor judgment timer (IJT) is initially set at zero, and starts counting up at the moment when GPS signals are lost. IJT keeps counting up as long as GPS signals are continuously lost, and returns to zero every time when signals become available again, if even for a moment. When IJT reaches a predefined time limit, the system judges the user as indoors. The time limit is decided according to the GPS-friendliness of the area: in rural areas it should be set to a low number, and in urban areas it should be a high number, because of the commonness of blocked signals. The outdoor judgment timer (OJT) works in almost the same manner as the IJT. When GPS signals become available even for an instant, the OJT starts counting up from zero. OJT keeps counting up as long as GPS signals are available, and returns to zero when signals are blocked. When OJT reaches the time limit, the system judges the user as outdoors. The time limit, like in the case of IJT, is predefined according to the area. In rural areas they are set high, and in urban areas they are set low, because there can be frequent signal blocks even when the user is outside.

The system determines that the user has visited a building when the user is first judged as indoors, then as outdoors. Then, the system records the current user location (latitude, longitude) and the approximate duration of the visit.

*3) Finding frequently visited shops:* From the record of users' visits to shops, we can reveal the presence of frequently visited shops, by searching for clusters of recorded visits. But since the locations of the recorded visits contain errors, the exact shops that the user is frequently visiting cannot be directly determined. Here, we propose a technique which automatically finds out the exact shops which the user is frequently visiting using an

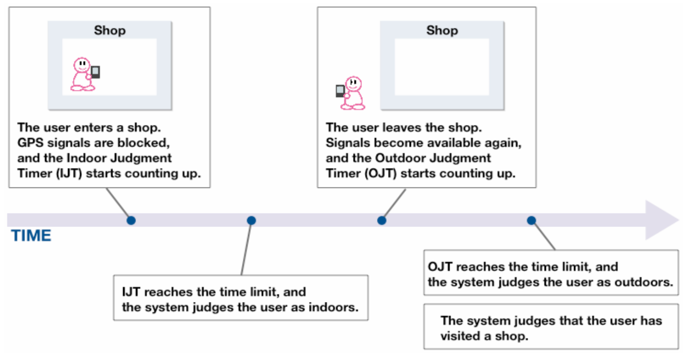

Shop
The user enters a shop.
GPS signals are blocked,
and the Indoor Judgment
Timer (IJT) starts counting up.
Shop
The user leaves the shop.
Signals become available again,
and the Outdoor Judgment
Timer (OJT) starts counting up.
TIME
IJT reaches the time limit, and
the system judges the user as indoors.
OJT reaches the time limit, and
the system judges the user as outdoors.
The system judges that the user has
visited a shop.

Fig. 4. Detecting visits to shops

estimation algorithm. The estimation algorithm is based on

*t*-test.

Whenever a new visit to a shop is detected, the system picks up shops that are located within a predefined radius from the location at which the visit took place. Then, for each of those shops, the system searches for past visits within a certain distance from the shop (*sample visits*). From the sample visits, the system evaluates if it is plausible that the shop is a frequently visited shop, by applying two tailed *t*-test to the sample visits. A more detailed explanation of this method is as follows. First, we assume that the observed latitude and longitude values of visits to a certain shop follow a normal distribution, with the actual location of the shop as the mean. Given this assumption, the latitude and longitude values of the sample visits around a shop will follow *t*-distributions. Then, we use *t*-test to evaluate if the means of the *t*-distributions can be regarded as *statistically identical* to the actual location of the shop. If the shop is a frequently visited shop, the two locations should be statistically identical. Fig. 5 illustrates the procedure of this method.

If, as the result of the *t*-test, the two locations can be considered as statistically identical, the shop is judged as a user's frequently visited shop. The shops is included in the user's frequently visited shops list, with a rating value calculated from the number of visits and the average duration of those visits.

If a large number of sample visits can be expected to be acquired, we can use Bayesian estimation instead of *t*-test. In this case, the *plausibility* that the shop is a frequently visited shop, is calculated using the following equation.

$$P = \iint_A p(\mu_x, \mu_y) \qquad (1)$$

$p(\mu_x \mu_y)$ is a probability density function calculated using Bayesian estimation. It indicates the probability density with which the shop that caused the sample visits is located at $(\mu_x, \mu_y)$. $A$ is a small area centered around the actual shop location. Simply put, $P$ is the estimated probability that the location of the shop at which the sample visits took place is located inside $A$. If the plausibility $P$ is above a threshold value, the shop is judged as the user's frequently visited shop. The shops are added to the user's frequently visited shops list,

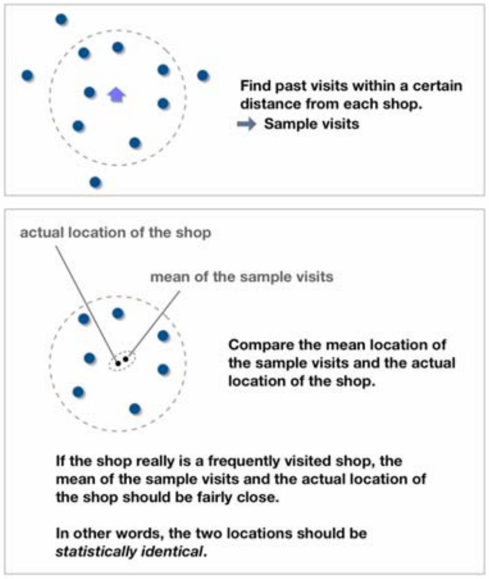

Find past visits within a certain distance from each shop.
Sample visits
actual location of the shop
mean of the sample visits
Compare the mean location of the sample visits and the actual location of the shop.
If the shop really is a frequently visited shop, the mean of the sample visits and the actual location of the shop should be fairly close.
In other words, the two locations should be statistically identical.

Fig. 5. Estimation Algorithm (*t*-test)

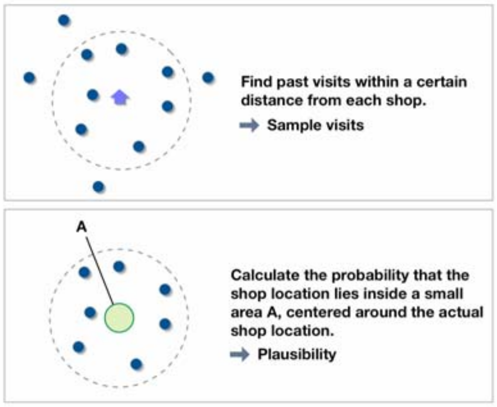

Find past visits within a certain distance from each shop.
Sample visits
A
Calculate the probability that the shop location lies inside a small area A, centered around the actual shop location.
Plausibility

Fig. 6. Estimation Algorithm (Bayesian estimation).

with rating values calculated from the plausibility $P$ and the average duration of the visits. Fig. 6 illustrates this method.

We have described two methods for finding frequently visited shops, one based on *t*-test, and the other based on Bayesian estimation. Whichever method we use, we end up with a list of frequently visited shops and rating values (Fig. 7). This list is stored in the server database and is used for recommendation.

## B. *Recommendation Phase*

Upon user request, the server recommends shops using the list obtained in the data acquisition phase. Recommendation is done in two steps: filtering, and adding weights according to areas.

*1) Filtering:* As the filtering algorithm, we use the item-based collaborative filtering algorithm. The similarity between two shops A and B is calculated using the following equation.

$$Sim(A,B) = \frac{\sum_u R_{u,A} R_{u,B}}{\sqrt{\sum_u {R_{u,A}}^2}\sqrt{\sum_u {R_{u,B}}^2}} \qquad (2)$$

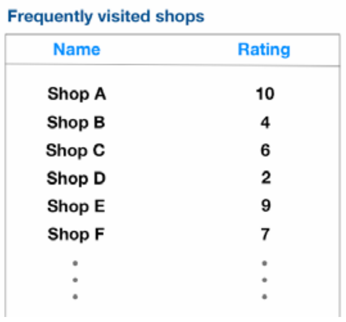
Here, $R_{u,A}$ indicates the rating value for shop $A$ by user $u$, and $R_{u,B}$ indicates the rating value for shop $B$ by user $u$. The similarity increases when there is an observed tendency that users who frequently visit shop $A$ also frequently visit shop $B$. Since item-based collaborative filtering does not take into account the content of the data, correlations between shops of different categories, such as cafes and clothing stores, can be defined. The system picks out several shops which have high similarity with the user's frequently visited shops. These shops are regarded as having high chances of matching the user's preferences well.

Calculating the similarities between shops require large computation cost, so the calculation cannot be done at the time of request, nor does it need to be.

The similarities reflect users' long-term shopping activities, and thus it can be reasonably assumed that their changes within a short amount of time are subtle. A cycle of once a day should be sufficient for our purposes.

*2) Adding weights according to areas:* Our system is intended to be used in the city, which means that there will be physical distances between users and recommended shops. Compared to online shopping where users can check recommended items with one click of a mouse, our system requires users to overcome these distances before they can appreciate the results of the recommendations. In cities like Tokyo where many people shop on foot, the distances can easily become too demanding for users. Therefore, we must make sure that the recommended shops are relatively easily accessible from the users.

To meet this requirement, we first divide the city into *areas*, and model users' movements using Markov models, with areas as nodes. The shops picked out by the filtering algorithm are added with weight values according to the areas in which they are located. Shops with higher weight values are given increased chances of being recommended to the user. Shops located in the current area of the user, or in areas where the user is likely to advance next are added large weight values, and thus will more likely to be recommended. Below we explain this procedure in detail.

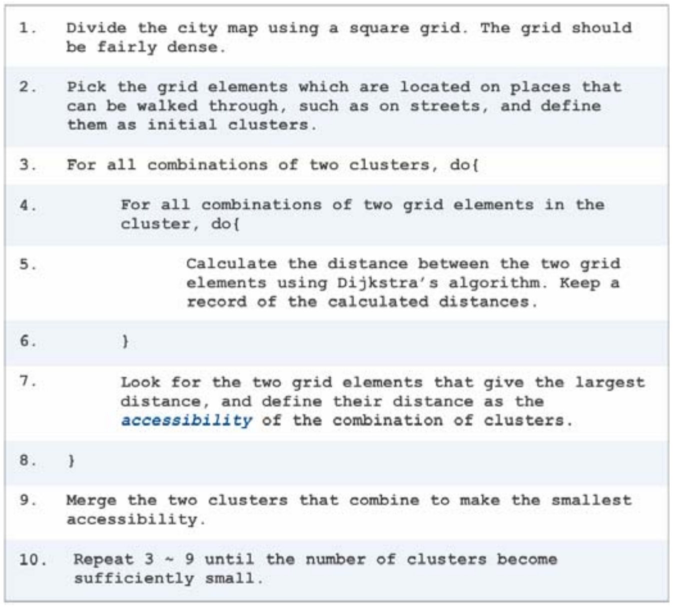


Frequently visited shops

| Name | Rating |
|---|---|
| Shop A | 10 |
| Shop B | 4 |
| Shop C | 6 |
| Shop D | 2 |
| Shop E | 9 |
| Shop F | 7 |
| ⋮ | ⋮ |

Fig. 7. List of frequently visited shops

**1. Dividing the city map into areas** Areas must be defined so that any two points located in the same area are easily accessible from one to the other. Our algorithm for this task, based on cluster analysis, is as follows:



```
1.  Divide the city map using a square grid. The grid should
    be fairly dense.
2.  Pick the grid elements which are located on places that
    can be walked through, such as on streets, and define
    them as initial clusters.
3.  For all combinations of two clusters, do{
4.      For all combinations of two grid elements in the
        cluster, do{
5.          Calculate the distance between the two grid
            elements using Dijkstra's algorithm. Keep a
            record of the calculated distances.
6.      }
7.      Look for the two grid elements that give the largest
        distance, and define their distance as the
        accessibility of the combination of clusters.
8.  }
9.  Merge the two clusters that combine to make the smallest
    accessibility.
10. Repeat 3 ~ 9 until the number of clusters become
    sufficiently small.
```

The resulting clusters are chosen as the areas. The algorithm defines areas so that the maximum distance the user has to cover to travel between two points in the same area is minimized. Fig. 8 illustrates how areas are defined. Areas should be redefined anytime there are significant changes in the city landscape, such as openings of new streets, etc.

**2. Modeling user movement** User movement is modeled as a first-order Markov model, with areas as nodes. As we have previously explained, the periodic location of the user is recorded by the system in the data acquisition phase. Transition probabilities between areas are calculated from this record. Higher transition probability indicates more chances of the user advancing to the area. The model should be periodically reconstructed, for example in a cycle of once a day.

Weight values are added to the shops chosen by the filtering algorithm, according to the areas in which they are located. Shops in the same area as the user, or areas with large transition probabilities from the current area are given the most weights. This way, the system takes into account the user's usual shopping routes, and the ease of access from the user to each shop. The way the areas were defined guarantees that shops located in the same area as the user are easily accessible from the user's current location. Areas with large transition probabilities indicate the next step of the user's usual shopping route, and shops located in those areas are likely to become easily accessible soon. The system picks up a few shops with the largest weights, and presents them to the user as the final

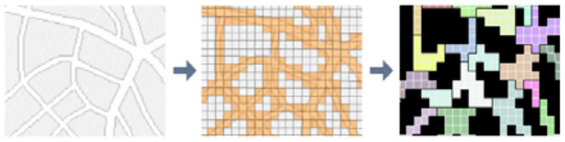


Fig. 8. Dividing the city map into areas

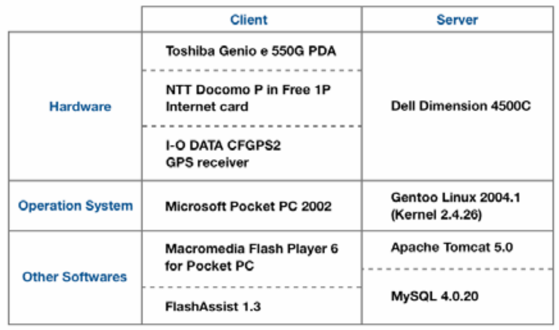
recommendation results.

Although not exploited in our current system, there are some other information that can be used, such as walking distance or the time of day. For example, we can put extra weights on restaurants when the recommendation was requested around noon, or put weights on cafes if the user has walked long distances. Another type of information that may be useful is time tables of movies or events. By combining these information with the user's location records, we can figure out what kinds of movies or events the user likes, and extend our system to be able to recommend data in these categories.

## IV. Implementation

Fig. 9 shows the platform for our implementation of CityVoyager. As the client device, we use a Toshiba Genio e 550G PDA (Fig. 10), with a GPS receiver and an Internet card. Since the memory and computational capabilities of our client device is rather weak, transforming location data into a list has to be done in the server, and raw location data must be sent to the server. As explained earlier, this generates significant privacy risks and thus it is intended only for evaluation purposes.

We have developed two custom applications for our system: the client application and the server application. The client application consists of the main application written in C++, and the user interface developed as a Macromedia Flash file. The server application consists of several Java Servlet files. The main client application and the user interface communicates data by sending and receiving XML sentences through TCP/IP sockets. The client application and the server application communicates through Dial-up Internet connection.



| | Client | Server |
|---|---|---|
| Hardware | Toshiba Genio e 550G PDA | Dell Dimension 4500C |
| | NTT Docomo P in Free 1P Internet card | |
| | I-O DATA CFGPS2 GPS receiver | |
| Operation System | Microsoft Pocket PC 2002 | Gentoo Linux 2004.1 (Kernel 2.4.26) |
| Other Softwares | Macromedia Flash Player 6 for Pocket PC | Apache Tomcat 5.0 |
| | FlashAssist 1.3 | MySQL 4.0.20 |

Fig. 9. Platform

Fig. 10. Hardware

SSH is used for all communication between the client and the server.

### A. Client Application

Here, we describe the functions of the main client application. The user interface will be explained later. The client application was developed using eMbedded Visual C++ 4.0.

The client application acquires the current location from the GPS receiver every second. Once per minute, it sends the current location to the server, to be used for modeling the user's behavior with Markov models.

The client application detects users' visits following the procedure discussed in the previous chapter. The indoor judgment timer (IJT) is set to 20 seconds, and the outdoor judgment timer (OJT) is set to 10 seconds. These values were determined after several field studies conducted at Daikanyama to observe how our GPS receiver functions in urban environments, and should be sufficiently effective for use in our system. But since a thorough study to define optimal values for these timers wasn't conducted, it is possible that there exists values which yield better detection accuracy than these values. Each time a visit is detected, the client application sends the location and the duration of the visit to the server application.

### B. Server Application

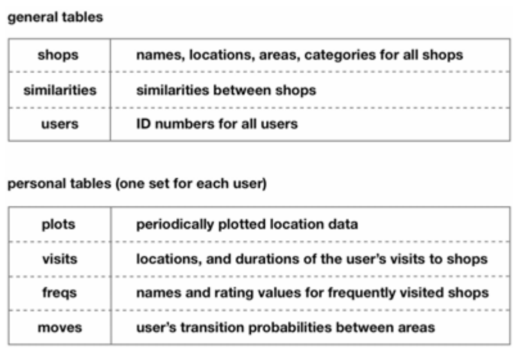
The server application is developed as several Java Servlet files. Its two main functions are finding each user's frequently visited shops, and making recommendations.

The server application finds the user's frequently visited shops with the estimation algorithm described in the previous section, based on a two-tailed *t*-test with a rejection region of 10%. Upon user request, the server application makes recommendations using the item-based collaborative filtering algorithm.

Fig. 11 lists the contents of the MySQL server database. The database consists of three general tables, and a set of four personal tables for each user. For example if there are ten users in the system, there will be 3+4*10=43 tables in the database.

### C. User Interface

The user interface of CityVoyager is developed as a Macromedia Flash file, using Macromedia Flash MX Professional 2004. We used FlashAssist to let the user interface be displayed in full screen mode. The main reason we made the interface separate from the client application is to gain the freedom to modify the user interface into a form that can be easily used by people not accustomed to manipulating PDAs. The Pocket PC is designed to be a general information processing machine, just like personal computers. An interface for a general information processing machine tends to require longer learning periods compared to an interface for a machine used for a specific objective. Considering that our system has to be evaluated by a lot of users, most of whom would have never touched a PDA, we decided to modify the interface in a way that the system could be easily used like a machine intended for a specific use.



general tables

| | |
|---|---|
| shops | names, locations, areas, categories for all shops |
| similarities | similarities between shops |
| users | ID numbers for all users |

personal tables (one set for each user)

| | |
|---|---|
| plots | periodically plotted location data |
| visits | locations, and durations of the user's visits to shops |
| freqs | names and rating values for frequently visited shops |
| moves | user's transition probabilities between areas |

Fig. 11. Server database

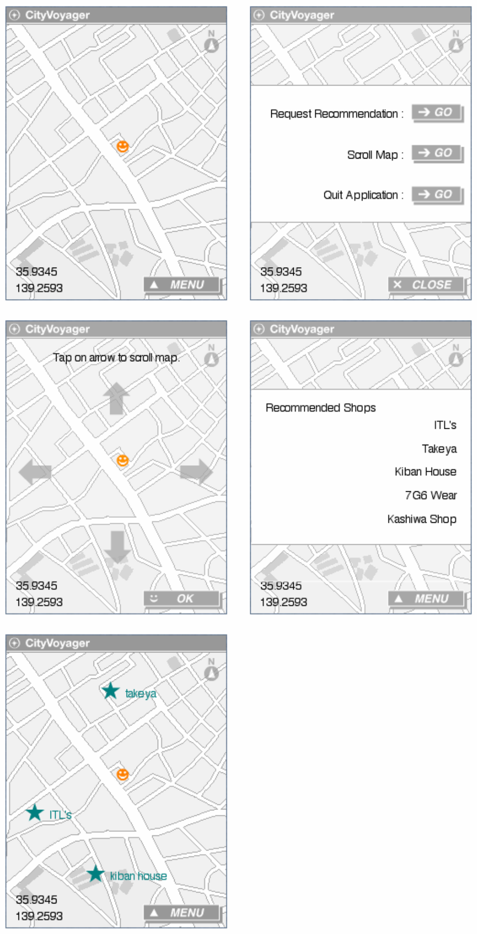


Fig. 12. Screenshots

The user interface consists of several screens (Fig. 12), each serving different functions to the user. The map screen is the main screen. The city map is displayed, with a face icon indicating the current user location. When the user touches the menu button in the lower right corner, the screen changes to the menu screen. Here, the user can execute three commands, "request recommendation", "scroll map", and "quit application". We intended to lessen the chance of accidentally quitting the application by only enabling the user to choose the quit option from the menu screen. In default Pocket PC applications, applications can be easily quit anytime by touching the close button in the upper right corner of the screen, which may result in many unintended quits for novice users. The scroll map screen appears when the user selects "scroll map" from the menu screen. Tapping on one of the arrows will scroll the map in the direction of the arrow. When the user chooses "request recommendation", recommended shops will appear in a list. The recommended shops will also be shown as star icons in the map screen.

## V. IMPLEMENTATION

We conducted an evaluation test of CityVoyager at Daikanyama, Tokyo. The location was chosen for its exceptional GPS-friendliness among Tokyo's many shopping districts. The evaluation test was carried out in two phases. In the first phase, we recorded long-term location data for a number of users and evaluated how well our system can find users' frequently visited shops. In the second phase, we evaluated the effectiveness of our recommendation results.

### A. *Data Acquisition Phase*

We asked 10 users (ages 18 to 25, 6 male and 4 female) to shop freely at Daikanyama, with our client devices in their bags. Due to the limited time available for the test, we asked users to visit as many shops as possible. Each test session lasted for approximately two and a half to three hours, and each user participated in at least one session. After the test, users were asked to report a list of shops that they frequently visited. This list was compared with the frequently visited shops that were automatically estimated by the system.

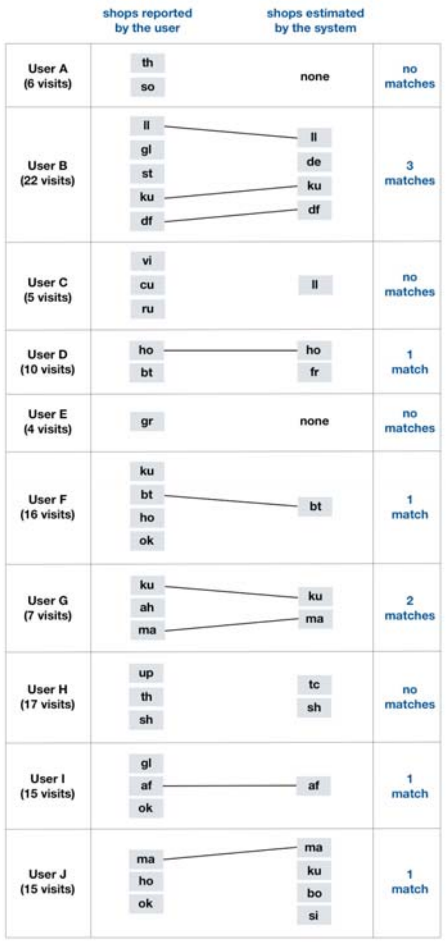
We requested users to limit visits to cafes or restaurants to only when they were extremely tired and needed rest. This was because visits to cafes or restaurants take up too much test time compared to other types of shops, and we had limited test time available.

Fig. 13 shows how the frequently visited shops estimated by the system compare with the shops actually reported by the user. We will discuss these results later.

### B. *Recommendation Phase*

To evaluate the effectiveness of our system, we made comparisons between our system and two other methods. One was shopping without the use of any guides, in which users were asked to follow their intuition and freely visit shops. The other was conventional location-aware recommendation, like the ones used in mobile phones, in which the shops closest to the user's current location were recommended.
We asked 2 users (age 24, male and age 25, female) to visit three shops for each of the above three methods, and give each shop a rating value in a scale of seven points. A scale of seven is commonly used in surveys, since it is known that reliabilities of subjective ratings do not increase dramatically with scales of more than seven.

Fig. 14 shows the results of the test. We will provide a detailed discussion of these results in the next section.



| | shops reported by the user | shops estimated by the system | |
|---|---|---|---|
| User A (6 visits) | th, so | none | no matches |
| User B (22 visits) | ll, gl, st, ku, df | ll, de, ku, df | 3 matches |
| User C (5 visits) | vi, cu, ru | ll | no matches |
| User D (10 visits) | ho, bt | ho, fr | 1 match |
| User E (4 visits) | gr | none | no matches |
| User F (16 visits) | ku, bt, ho, ok | bt | 1 match |
| User G (7 visits) | ku, ah, ma | ku, ma | 2 matches |
| User H (17 visits) | up, th, sh | tc, sh | no matches |
| User I (15 visits) | gl, af, ok | af | 1 match |
| User J (15 visits) | ma, ho, ok | ma, ku, bo, si | 1 match |

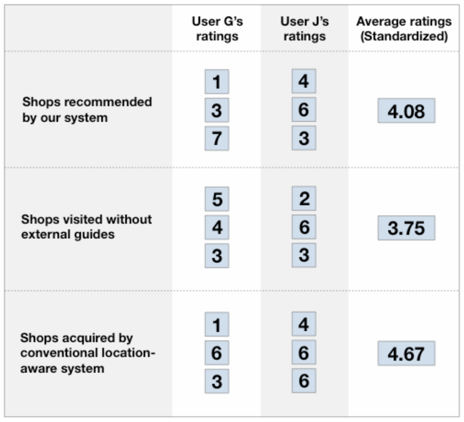
Fig. 13. Results of data acquisition phase

## VI. Discussion

### A. *Data Acquisition Phase*

In the data acquisition phase, we compared the frequently visited shops estimated by the system, and the shops actually reported by the users. A total of 17 shops were found by the system, and of those shops 9 proved to be actual frequently visited shops reported by the users. In other words, the system gave false positives with a rate of 47%. Also, the total number of frequently visited shops reported by the users was 29. Comparing this with the number of correctly estimated shops, which was 9, we can see that only 31% of the frequently visited shops reported by the users have been successfully detected by the system.

The most reasonable explanation for these unsatisfactory results is the sheer lack of data. Of the 10 users, only 5 have recorded more than 10 visits. This is too low, even considering the severely limited test time. As we had not closely monitored the users during the evaluation test, we do not know what caused this inadequate detection of visits. One likely reason, is that as we urged users to hurry and to visit as many shops as possible, users did not spend enough time inside the shops for the indoor judgment timer (IJT) to reach the time limit. We may have had to adjust the time limit to accommodate for the limited time available for our test.



| | User G's ratings | User J's ratings | Average ratings (Standardized) |
|---|---|---|---|
| Shops recommended by our system | 1, 3, 7 | 4, 6, 3 | 4.08 |
| Shops visited without external guides | 5, 4, 3 | 2, 6, 3 | 3.75 |
| Shops acquired by conventional location-aware system | 1, 6, 3 | 4, 6, 6 | 4.67 |

Fig. 13. Results of recommendation phase

Another possibility is that users often visited shops that are either GPS-transparent, or are contained inside other large buildings. Visits to these types of shops cannot be detected by our method.

The results were disappointing, but there is evidence that hints the potential effectiveness of our system. If we look at users who had recorded more than 15 visits (user B, user F, user H, user I), we can see that the system has successfully detected 47% (7 out of 15) of their reported shops, and the rate of false positives was as low as 13%. The detection rate is still acceptable at best, but the false positives rate is impressive and good enough for actual use. This suggests that the accuracy of our method increases with the amount of data, and if sufficient data can be acquired, our system may be able to produce pleasing results. As for the rather low detection rate, there is one possible explanation: in our system, the estimation algorithm will inevitably have trouble when a user frequently visits two closely neighboring shops. The chances are, the system will only be able to detect one of them as a frequently visited shop. As we have conducted our test in a relatively small area crowded with shops, situations like this may have often occurred.

### B. *Recommendation Phase*

The results for the recommendation phase test show that, the average rating value for shops recommended by our system is higher than that for shopping without external guides, but lower than that for conventional location-aware methods, where shops closest to the user were recommended. But the fact that there are differences in average ratings does not by itself prove if our system is superior or inferior to the other methods. We need to check if the differences evident in the results are *statistically significant*. To check this, we used a combination of a two-tailed *F*-test and a two-tailed *t*-test. In either test, we set the rejection region to 10%. The results of the tests showed that the differences observed in the average ratings were not statistically significant, and thus we could not statistically prove the superiority (or inferiority) of our system. Obviously, the lack of users and test time is responsible for these disappointing results.

Let us discuss the advantages of using our system from a non-statistical point of view. Shopping without external guides may be effective for finding good shops, but only if there is plenty of time. In a city like Tokyo where there exists at the same time a vast number of shops and ridiculously complicated streets, finding shops that match individual tastes will require much time and effort. Using CityVoyager can help in such situations by narrowing down the search space. Conventional location-aware methods give shops that are closest to the user's current position. In areas like Daikanyama, there will be always be countless shops within walking distance, and presenting shops close to the user will not be so much of a help. CityVoyager should be able to make more effective recommendations in such cases, since CityVoyager picks out shops according to personal tastes and preferences, not just geographical conditions.

## VII. Conclusion and Future Work

In this paper, we have proposed CityVoyager, a shop recommendation system for real-world shopping based on user location history. The results of the evaluation test show that CityVoyager estimates users' frequently visited shops with acceptable accuracy when there is a sufficient amount of data, and also demonstrate its potential ability to provide users with beneficial recommendations.

Since the amount of data collected in our evaluation test was too small for statistical analysis, we must conduct another evaluation test in a larger scale. We are planning of porting CityVoyager to mobile phones and making it publicly available, which should help us gather users for our evaluation test.

One modification which should definitely be done in future versions of CityVoyager is introducing some constraints to the recommendation results regarding some basic information about the user, such as age or sex. In the evaluation test, a shop which only sells fashion items for women was recommended by our system to a male user, and consequently received a rating of 1. Obvious mismatches like this should be excluded from the recommendation results, by defining several basic attributes for each shop and filtering out data with unwanted attributes.

## References

[1] Fogg, B. J., Persuasive Technology : Using Computers to Change What We Think and Do. Morgan Kaufmann Publishers, 2003.

[2] Want, R., Hopper, A., Falcao, V., and Gibbons, J. The Active Badge Location System. ACM Transactions on Information Systems (TOIS), v.10 n.1, p.91-102, Jan. 1992.

[3] Want, R., Schilit, B., Adams, A., Gold, R., Petersen, K., Goldberg, D., Ellis, J., and Weiser, M. The ParcTab Ubiquitous Computing Experiment. Technical Report CSL-95-1, Xerox Palo Alto Research Center, March 1995.

[4] Bahl, P., and Padmanabhan, V. N. RADAR: An In-Building RF Based User Location and Tracking System. In INFOCOM 2000, pages 775– 784, 2000.

[5] Bahl, P., and Padmanabhan, V. N. A Software System for Locating Mobile Users: Design, Evaluation, and Lessons. MSR Technical Report, February 2000.

[6] Ward, A., Jones. A., Hopper, A. A New Location Technique for the Active Office. In IEEE Personal Communications, Vol. 4, No. 5, October 1997, pp 42-47.

[7] Aboud, G.D., Atkeson, C.G., Hong, J., Long, S., Kooper,R., Pinkerton, M., Cyberguide: A mobile context-aware tourguide, ACM Wireless Networks, 1997, pp.421-433.

[8] Marmasse, N. Schmandt, C. Location-aware Information Delivery with Commotion. In Proceedings of the 2nd international symposium on Handheld and Ubiquitous Computing table of contents Bristol, UK. Pages: 157 - 171, 2000.

[9] Ashbrook, D. and Starner, T. Learning signifcant locations ?nd predicting user movement with GPS. In Proc. of 6th IEEE Intl. Symp. on Wearable Computers, Seattle, WA, 2002.

[10] Zhou, C., Frankowski, D., Ludford, P., Shekhar, S., Terveen, L. Discovering Personal Gazetteers: An Interactive Clustering Approach. In Proceedings of ACM GIS 2004, Washington DC, 2004, pp. 266-273.

[11] Chen, L., and Sycara, K. WebMate: Personal Agent for Browsing and Searching. Proceedings of the 2nd International Conference on Autonomous Agents and Multi Agent Systems, AGENTS ’98, ACM, May, 1998, pp. 132 - 139.

[12] ] Lang, K. NewsWeeder: Learning to Filter Netnews. In Proceedings of 12 th International Conference on Machine Learning, Lake Tahoe, CA, Morgan Kaufmann, pp. 331-339, 1995.

[13] Goldberg, D., Nichols, D., Oki, Brian M., Terry, D., Using Collaborative Filtering to Weave an Information Tapestry. Communications of the ACM, v.35 n.12, p.61-70, Dec. 1992.

[14] Shardanand, U., Maes, P., Social Information Filtering: Algorithms for Automating ”word of mouth”. In Proceedings of ACM CHI’95 Conference on Human Factors in Computing Systems. NewYork, (pp. 210-217).

[15] Sarwar, B., Karypis, G., Konstan, J., Riedl, J Item-based Collaborative Filtering Recommendation Algorithms. In Proceedings of 10th International World Wide Web Conference, ACM Press, 2001, pp. 285-295.

[16] Asthana, A., Cravatts, M., and Krzyzanouwski, P. An Indoor Wireless System for Personalized Shopping Assistance. In Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, pp. 69-74, Santa Cruz, California, December 1994. IEEE Computer Society Press.

[17] Brunato, M., Battiti, R., Villani, A., Delai, A. A Location-Dependent Recommender System for the Web. Technical Report DIT-02-093, Informatica e Telecomunicazioni, University of Trento. 2002.

**Yuichiro Takeuchi** received a B.S. in Engineering from Department of Systems Innovation in 2003, and M.S. in Frontier Informatics from Department of Frontier Informatics in 2005, at The University of Tokyo, Japan

Currently he is enrolled as a Ph.D candidate at Department of Frontier Informatics, The University of Tokyo, Japan. His main fields of research are context-aware computing and human-computer interaction.

**Masanori Sugimoto** became a Member (M) of the IEEE in 1995. He received B.Eng. and M.Eng. degrees from Department of Aeronautics and Astronautics, and Dr.Eng. degree from Interdisciplinary Course on Advanced Science and Technology, The University of Tokyo, Japan, in 1990, 1992, and 1995, respectively.

In 1995, he joined Research and Department Development, National Center for Science Information Systems (currently National Institute of Informatics), Tokyo, Japan. He became an associate professor in Information Technology Center, University of Tokyo, in August 1999. Currently, he is an associate professor in Department of Frontier Informatics, Graduate School of Frontier Sciences, University of Tokyo. He was a visiting scientist in Department of Computer Science, University of Colorado, Boulder CO, USA (from January 1997 to November 1997). His research concern is related to human-computer interaction, especially mixed reality, mobile computing, computer supported collaborative work/learning, intelligent systems, and so on.

Dr. Sugimoto is a member of IEEE, ACM, ISLS, IEICE, JSAI, IPSJ, and JCSS.