Design of the Recommendation Module for Context Aware VOD Museum Guide Android App

Jaegeol Yim

Department of Computer Engineering, Dongguk University at Gyeongju, Gyeongbuk 780-714, Korea
yim@dongguk.ac.kr

Abstract

Location based service such as automobile navigation system is one of the most valuable commercial products. As the buildings, factories, and shopping malls are getting bigger and bigger, demand for indoor location based service is also increasing rapidly. In addition to user's location, the context of the usage such as user's personal information including age, hobby, occupation, user's emotion and so on is also important factor that determines the most proper service. Smartphones are best suited platform for context aware service implementation because the owner of a smartphone always carries it and saves personal information in it. Besides, record of smartphone use history that is automatically saved in the smartphone is a good resource from which we can infer the context. Furthermore, smartphones are equipped with powerful processing unit and pretty accurate sensors. Making use of the features of smartphones, we have developed a context aware museum guide Android app. It is also a video-on-demand app because it selects best suited video based on the user's context and plays the video. To the best of our knowledge, our app is the first indoor-location-based, context-aware, and video-on-demand Android app. This paper describes the recommendation module of it in detail.

Keywords: Smartphone, Indoor Location Based, Context Awareness, VoD, App

1. Introduction

One of the most popular buzzwords is convergence. As science and technology evolve, computers become smaller and smaller while their computing power become stronger and stronger. At the same time, sensor devices also become smaller and smaller while their accuracy become higher and higher. Finally, they all converged into a telephone to become a smartphone which is one of the best examples of successful convergence. Nowadays, a smart phone, convergence of computer, telephone and many sensors, is one of the most common personal belongings.

As a smart phone is equipped with a computer and many sensors, various useful applications running on it have been developed. Since a smart phone is one of the most common personal belongings, it is one of the most appropriate devices which provide location based and context aware services.

Among the most popular mobile operating systems (Android, Blackberry, iOS, and Windows Phone 7), Android is most attractive to us because it is easiest to access and to start developing an app running on it. This paper introduces our indoor location based, context aware and video on demand Android application that is useful to visitors of a museum.

In a museum, there are many showcases in which one or more exhibits are displayed. Our Android app actively recommends showcases that the user most likely wants to visit. Our design of the recommendation module is described in detail in this paper.
2. Related Works

Our Android app is an indoor location based, context aware, and video on demand system. Researches on location based service (LBS) started relatively long ago and practical LBS devices are around people's everyday life, nowadays. Without accurate knowledge of the geographical position of a user, development of LBS system is not possible. The authors of [1] proposed the linear matrix inequality method that computes the optimum value of the range intersection where the user lies. The authors of [2] proposed an adaptive extended Kalman filter moving target tracking algorithm using the measurement signals of time difference of arrival and the frequency difference of arrival.

There are so many location based applications. The authors of [3] introduced a network based ticket reservation system which is a location-based smartphone application with augmented reality. The authors of [4] introduced Personalized Location-based Traveler Recommender System that provides personalized tourism information to its users.

As the cities are filled with huge buildings and numerous large stores are open underground, most of the people spend most of their time indoors. Therefore, LBS should be available not only outdoor but also indoor. Developing outdoor LBS is much easier than developing indoor LBS (ILBS) because of GPS (Global Positioning System) and map providers (Google, for example). Since we cannot use GPS or the map in ILBS development, researches on indoor positioning and floor map (or drawing) provide service [5-8] are actively being performed.

A context ware service determines the situation of the user by investigating sensor values gathered from the sensors equipped on the smart phone (GPS, temperature, humidity, brightness, compass, accelerometer), the history of using the phone (call, SMS, app), personal information (images, scheduler, addresses) and the specification of the device (size of the monitor, speed of the CPU, size of the memory) and infers the content which seems most likely useful to the user [10-14]. Video on Demand (VoD) are systems that allow users to select and watch video content [15-17].

We adapt the K-Nearest Neighbor (K-NN) method introduced in [18] to implement our indoor positioning module. The deployment of K-NN consists of two phases, off-line phase and on-line phase. During the off-line phase, a look-up table is built. The entire area is covered by a rectangular grid of points called candidate points. At each of candidate points we collect the received signal strength indices (RSSI) many times. Let RSSI_{i,j} denote the j-th received signal strength index of the signal sent by i-th AP. A row of the look-up 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 the associated candidate point. A list of signal strengths consists of five integers, RSSI_{1}, RSSI_{2}, ..., where RSSI_{i} is an average of signal strengths RSSI_{i,j} received at (x, y) and sent by i-th AP.

In the second phase, or on-line phase, the positioning program gathers RSSIs the user receives at the moment. Let X be the vector of the collected RSSIs. K-NN, then finds the candidate point that is closest to X from the look-up table, and returns it as the user’s current location. If K equals 2, then it will find two closest candidate points and return the average of them as the user’s current location.

The VOD system consists of a video streaming server, database server and an Android application (client) as shown in Figure 1. A user of VOD application specifies the video he wants to watch on the user interface of the application. Then, the database server searches for the video file that matches the requested video and sends back the file path with the URL of the streaming server to the mobile client. If the mobile application accesses the streaming server then the streaming server streams out the video to the client. The media player at the client saves and plays the video as it receives the video stream.
The main topic of this paper is describing the recommendation module of the app in detail. The recommendation module figures out a list of showcases that the visitor would most likely visits. Then it shows a route for the visitor. The visitor will most efficiently visits all the recommended showcases if he or she follows the route. That is, the route is a shortest path that covers all the recommended showcases.

Shortest path recalls us Dijkstra's algorithm that solves the single-source shortest path problem. For a given source vertex, this algorithm finds the shortest path between the source and every other vertex. This algorithm can also be used to find the shortest path between two given vertices. Another famous algorithm that finds shortest paths is Floyd-Warshall algorithm. This algorithm finds the shortest paths between all pairs of vertices. These two algorithms are efficient. Dijkstra's algorithm takes \(O(n^2)\) and Floyd-Warshall algorithm takes \(O(n^3)\) time where \(n\) represents the number of vertices.

However, recommending or finding the most efficient route for the visitor cannot be solved by Dijkstra's or Floyd-Warshall algorithms. Given a set of recommended showcases, finding the most efficient route can be interpreted as the travelling sales person problem that is a well-known NP-hard problem. Given a set of cities and distances between every pair of the cities, the sales person wants to find the shortest path to visit all the cities exactly once and come back home. The simplest solution of this problem enumerates all feasible routes that visit all the given cities exactly once. There are about \(n!\) such routes. For each of the feasible route, the solution calculates the cost of the route where cost of a route is the sum of the distances associated with the edges consisting of the route. NP-hard implies that so many possible solutions for the sales person problem has been proposed, but none of them is more efficient than the simplest solution in the worst case.

3. Functional Requirements

As an indoor-location-based, context-aware, and video-on-demand smartphone application, our Museum Guide has to detect whether the user is moving or standing still to watch an exhibit. Once the user is determined to be watching an exhibit, Museum Guide has to identify the exhibit being watched by the user and play the video which is mostly relative to the exhibit.
On top of these functions, it is desired to be able to estimate user's available time, or the time period for which the user is intending to spend in the museum. It is also desired to correctly guess the showcases (exhibits) that the user would like to visit (watch). Once we know the available time and the showcases (exhibits) most likely to be visited (watched), we can make most appropriate recommendations to the user. The following is a list of main functions of our Museum Guide:

- Detects whether the user is moving or not
- Detects whether the user is watching at an exhibit or not
- Recommends showcases which the user most likely wants to visit
- Recommends videos which the user most likely wants to watch
- Plays the video selected by the user
- Records "who and when watches which video" and manages history
- Records "who rated which video what level" and manages rating data

4. Design

The first function to be realized is detecting whether the user is moving or not. Among the available sensor values, y-axis accelerometer values best reflects the moving status. We collect y-axis accelerometer value every 50 milliseconds. By investigating 20 recently gathered y-axis accelerometer values we determine whether the smart phone is moving or not. When the user does not move, the variance of y-axis accelerometer values is small.

![Diagram of detecting moving status]

**Figure 2. The process of detecting moving status**

Let Y-ACC is an array of float numbers. We collect y-axis accelerometer values for 1 second and add them into Y-ACC in the order of collection. If the standard deviation of the values in Y-ACC is less than a certain threshold, th, then we conclude that the user is not moving. Our strategy is represented in Figure 2.
The user aims at the exhibit with his or her phone camera when the user is interested in the exhibit. This situation can be detected by investigating pitch value. When the phone is in straight portrait orientation, the value of the pitch is about -90. Therefore, if $|\text{pitch}+90|<30$ then we can conclude that the smart phone is in straight portrait orientation. Furthermore, if the phone does not move and $|\text{pitch}+90|<30$ then we conclude that the user is watching an exhibit.

For our showcase recommendation, referring to the Movielens [19] dataset, we designed our database as shown in Figure 3. It consists of the following tables: User, Occupations,
Ages, showcases, items, showcase-item, and visit. These tables are corresponding to the
dataset in Movielens. Note that duration-time attribute of showcase table represents the
average time a visitor stays at the showcase. Our showcase recommendation module finds the
entries (similar subscribers) which are the most similar to the user (the owner of the smart
phone). Then, it returns the showcases that have been most frequently watched by the similar
subscribers.

The procedure shown in Figure 4 leverages the set of subscribers who are similar to the
user. The procedure shown in Figure 5 finds such a set. As is shown in Figure 3, the attributes
of user table include occupation, hobby, birth date, and zip-code. If a subscriber has exactly
the same values for all the attributes as the user has, then we assume that the subscriber is
similar to the user. If we have found enough number of subscribers who are similar to the user,
then we stop the procedure. Otherwise, we find subscribers whose gender, hobby, occupation,
and age values are the same as those of the user and let them be similar-users. After that, if
the number of elements of similar-users is not big enough, then we find any subscriber whose
value of any attribute is the same as the value of the user and add the subscriber to similar-
users. In Figure 5, the ellipsis represents "Add subscribers whose occupation is the same as
that of the user to similar-users," and so on.

![Figure 5. The process of finding subscribers who are similar to the user](image-url)
Figure 6. The process to find the user’s available time

Figure 7. The process of producing a list of recommended showcases
The user would want visit the showcases that similar subscribers most visited, i.e. the elements of listShowcaseID as many as the user's available time allows. Therefore, we need to figure out the user's available time. We assume that the user's schedule is written in the calendar of the Android smartphone. So, we can find the user's available time by subtracting the current time from the next schedule as shown in Figure 6.

We have the user’s available time and a list of recommended showcases at hand by now. It is obvious that the user wants to visit showcases as many as possible. Each of the showcases is associated with duration that is a period-of-time a visitor usually spends there. Therefore, we recommend a showcase with shorter duration first. So, our showcase-recommend module sorts the entries of listShowcaseID in descending order of duration at the first step as shown in Figure 7. Then, it repeats deleting the first entry of listShowcaseID, adding the deleted entry to recommended showcases, and subtracting the duration associated it from available-time as long as available-time is positive. The user would leave the last showcase in the middle of visiting. For example, let available time be 100 time units, recommended showcases be showcase 1, 2, and 3, and the durations be 40, 50, and 60, respectively. The user would spend 40 and 50 time units at showcase 1 and 2, respectively and leave showcase 3, 10 time units after arriving there.

After obtaining recommended showcases, the procedure shows the optimal route the user should take in order to visit them all. Assuming that the showcases are located around the wall of the room as shown in Figure 8, the optimal route will visit the showcases in the clockwise or counter clockwise direction.

![Figure 8. An example exhibition room](image)

Theorem 1: Given an exhibition room configuration where the showcases are located along the walls, the optimal route that visits all the showcases is the one that visits the showcases in the direction of clockwise or counter clockwise.

<Proof> In this proof, distance(v_i, v_j) represents the distance from showcase v_i to showcase v_j. Let v_1, v_2, ..., v_n be the sequence of showcases representing the route that visits the showcases in the clockwise direction. Suppose there is an optimal route visiting showcases not clockwise direction, v_{i_1}, v_{i_2}, ..., v_{i_{n'}}, where the positions of v_i and v_{i+j} are changed. Since the distance(v_{i-1}, v_{i+j}) + distance(v_{i+j}, v_{i+1}) + distance(v_{i-1}, v_i) + distance(v_i, v_{i+j+1}) is
greater than distance\(v_i - 1, v_i\) + distance\(v_i, v_{i+1}\) + distance\(v_{i+1}, v_{i+2}\) + distance\(v_{i+2}, v_{i+3}\), the sequence \(v_1, v_2, ..., v_n\) would also be an optimal route.

The proof of Theorem 1 is also applicable to a subsequence of the sequence of showcases representing the route that visits the showcases in the clockwise direction \(v_1, v_2, ..., v_n\). Therefore, given any subset of the showcases, the optimal route is the one that visits the elements of the subset in the direction of clockwise or counter clockwise. This is stated in Theorem 2.

Theorem 2: Given an exhibition room configuration where the showcases are located along the walls, the optimal route that visits any subset of the showcases is the one that visits the elements of the subset in the direction of clockwise or counter clockwise.

Example 1: Let the content of the calendar be as shown in Table 1, the current time be 11:20, listShowcaseID be 1, 2, 4, 6, 8, 10, and the content of showcases table for them be as shown in Table 2. The procedure shown in Figure 7 finds that the available time is 40 minutes, rearranges the list into 8, 2, 3, 1, 4, 10, 6 based on the duration values of showcases, then it will return 8, 2, 3, 1, 4 as recommended showcases. The recommendation module will finally render the floor map and draw the optimal route 1, 2, 3, 4, 8 on the smart phone window.

<table>
<thead>
<tr>
<th>Start time</th>
<th>End time</th>
<th>place</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:00</td>
<td>13:00</td>
<td>cafeteria</td>
<td>Lunch with Sol</td>
</tr>
<tr>
<td>14:00</td>
<td>15:15</td>
<td>Lecture Room 201</td>
<td>Computer programming 101</td>
</tr>
<tr>
<td>15:30</td>
<td>16:45</td>
<td>Lecture Room 202</td>
<td>Calculus 101</td>
</tr>
<tr>
<td>18:08</td>
<td>18:10</td>
<td>Bus Stop</td>
<td>Leaves a commuter bus</td>
</tr>
</tbody>
</table>

Table 1. An example calendar

<table>
<thead>
<tr>
<th>ID</th>
<th>Duration</th>
<th>name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>Bronze Knives</td>
<td>Excavated from King Namool's tomb</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>Bronze accessories</td>
<td>Excavated from King Namool's tomb</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>Bronze harness</td>
<td>Excavated from King Namool's tomb</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>Patternless earthenware</td>
<td>Excavated from Jeongok-Li</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td>Chipped stone tools</td>
<td>Excavated from Jeongok-Li</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>Polished stone tools</td>
<td>Excavated from Goryong-Li</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>Gold accessories</td>
<td>Excavated from the Great King's tomb</td>
</tr>
</tbody>
</table>

Table 2. A part of showcases table

5. Implementation and Experiments

Recall that we investigated 20 recently collected y-axis accelerometer values and if the standard deviation of them is less than the threshold then we conclude that the user is not moving. Our experiments showed that the program perfectly correctly detects not moving state when the threshold is 0.5. Figure 9 shows example y-axis accelerometer values we have collected.

In Figure 3, we can see a few database tables where we record information needed for showcase recommendation. Each of the tables, we implemented web methods such as doGet for retrieving, doPost for updating, and others for other tasks as shown in Figure 10.
Our showcase recommendation procedure estimates available time by subtracting the current time from the start time of the next schedule. We obtain the current time with the following sentence. As the result of the instruction, the current time will be assigned to now.

```java
long now = System.currentTimeMillis();
```

The user's schedule is recorded in the calendar. In order to read from and write into calendar, we have to specify the following sentences on AndroidManifest.xml.

```xml
<uses-permission android:name="android.permission.WRITE_CALENDAR"/>
<uses-permission android:name="android.permission.READ_CALENDAR"/>
```

In order to read the schedules, we use context.getContentResolver as follows:

```java
Uri calendars = Uri.parse("content://com.android.calendar/events");
String[] projection = new String[] { "dtstart", "dtend" };
Cursor managedCursor = context.getContentResolver().query(calendars, projection, "dtstart">"+now, null, "dtstart asc") ;
```

We have implemented two recommendation modules, one for showcases and the other is for videos. Because of the space limitation, the former only is discussed. The process of it is described in Figure 5. In Figure 5, similar-users represents the subscribers who are similar to the user. We consider two persons are the most similar if all the attribute values are the same. We consider two persons are the second most similar if the values of gender, age, hobby and occupation are the same. We consider two persons are the third most similar if they share the same hobby. We make listShowcaseID with the showcases that have been watched by the members of Group. At this moment, listShowcaseID might be empty or too short. Therefore, we append the most popular showcase to the list if it is not in the list. By this time, the remaining of the process should be clear and we omit explaining it.
Our recommendation module takes the available time into account. That is, by investigating the current time and the scheduler of smart phone, it estimates the available time for which the user may spend in the museum. Based on the available time, it recommends appropriate number of showcases.

Our video player is interactive in that if the user touches an object on the screen then the player pauses playing the current video and starts playing another video that is associated with the touched object.

Figure 11 is a screenshot of showcase recommendation. From users table, Museum Guide finds subscribers who are most similar to the user. Then, it finds the showcases which have been most frequently visited by the similar users and call them most likely visited showcases. Finally, considering user's available time, it selects appropriate number of showcases from the most likely visited showcases and displays the selected ones. When user's available time is longer than the total time to visit all the most likely visited showcases, it selects most popular showcases.

![Figure 11. A screenshot of showcase recommendation](image1)

Figure 11 shows the behavior of the interactive player. In the middle of playing the video, the user touched the pagoda as shown in the first screenshot. Our player recognized that the pagoda was touched as shown in the second screenshot. It is now supposed to start playing the video associated with the pagoda. The third screenshot shows that there is no video associated with the pagoda.

![Figure 12. Screenshots of our interactive player](image2)
6. Conclusions

We have introduced our design of the recommendation showcases of our Museum Guide in detail. To the best knowledge of ours, it is the first indoor-location-based, context-aware, and video-on-demand smartphone app. The indoor positioning process we have implemented in this research is not accurate enough to pinpoint where the user is. Therefore, we are now developing dead reckoning indoor positioning module using the sensors installed on smartphones.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012-0002611). This work (Grants No. C0033172) was also supported by Business for Cooperative R&D between Industry, Academy, and Research Institute funded Korea Small and Medium Business Administration in 2012.

References

Author

Jaegeol Yim received the M.S. and Ph.D. degrees in Computer Science from the University of Illinois at Chicago, in 1987 and 1990, respectively. He is a Professor in the Department of Computer Engineering at Dongguk University at Gyeongju Korea. His current research interests include Petri net theory and its applications, Location Based Service, AI systems, and multimedia systems. He has published more than 50 journal papers, 100 conference papers (mostly written in Korean Language), and several undergraduate textbooks.