International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 11 (2015) pp. 27621-27633

http://www.ripublication.com

Design of the Location Based Service System for Local Area Tourism

Jaegeol Yim

Dept. of Computer Engineering, Dongguk University
at Gyeongju, Gyeongbuk Korea
E-mail: yim@dongguk.ac.kr

Abstract

For tourism, a location-based service (LBS) is extremely important because tourists are moving around to find attractions and places where they can eat, sleep, and drink. Since tourists are visiting places they are not familiar with, they cannot help but rely on LBS applications in order to obtain information they need. After reviewing recently introduced LBS systems, the design of an LBS system for local area tourism is discussed. One of the distinguishing features of the system is that live broadcast and online shopping are integrated into the system.

Keywords*:* Location Based Service, Online Shopping, Mobile Devices, Mobile Apps.

Introduction

Location Based Services (LBS) are so popular that they gave become a part of our daily lives. We drive without worrying about how to find a place, even when we visit it for the first time, using vehicle navigation systems running on smartphones. Examples of LBSs include vehicle navigation systems, smart museum guides, smart advertisements, travel information services, and so on [1].

This paper reviews techniques recently published in the LBS field. Similarity analysis between two trajectories, group nearest neighbors and content recommendation are widely used in LBS development. Recently introduced techniques of these topics are discussed [1].

Then, the design for an LBS system for local area tourism is introduced. The system displays a map. Then it also displays marks representing points of interesting within the boundaries of the map. It provides further detailed information about places associated with each mark selected by the user [1].

Related works

A trajectory is a sequence of ordered pairs, (time, coordinates), representing a moving object located at the place represented by the coordinates at that time indicated. With similarity between trajectories, one can predict an object's movement, associate a trajectory with the road network, analyze traffic flow, and find the migration patterns of wildlife. Therefore, the similarity problem has been a hot research topic in various fields. Chen et al. [2] introduced a dynamic time warping algorithm, shown in Figure 1, to estimate similarity [1].

DTW(*Q*, *C*, *D*)// *Q* and *C* are the two trajectories. *D* is the distance matrix, each *d*[i][j] in *D* is the distance between q_i and *c*

1 dtw[]= new double [n][m];// initialize matrix dtw
2 dtw[0][0]= 0;
3 for i = 1: n;
4 dtw[i][1]= dtw[i-1][1] + d[i][1];
5 for j= 1:m;
6 dtw[1][j]= dtw[1][j-1] + d[1][j];
7 for i = 1: n;
8 for j= 1: m;
9 dtw[i][j]= d[i][j]+min{dtw[i-1][j-1], dtw[i-1][j], dtw[i][j-1]}
10 return dtw;

Figure 1: The process of the Dynamic Time Warping algorithm [1, 2].

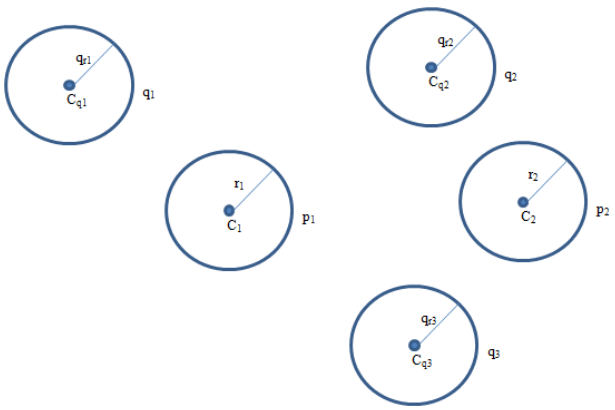

Figure 2: A range-based probabilistic group nearest neighbor query [1, 2]

Group nearest neighbor search has been widely used in solving practical problems in computer science. Finding the point in P={p1, p2} that is nearest to all points in Q={q1, q2, q3} in Figure 2 can be an example of a group nearest neighbor search if the points in P and Q are fixed. If the locations of points in P and Q are not fixed but any place within range with an arbitrary distribution, then the query becomes a range-based probabilistic group nearest neighbor (RP-GNN) query. Chen

et al. [3] proposed two novel pruning methods to improve the performance of RP-GNN [1].

Seo and Ahn [4] proposed a hybrid content recommendation service using LBS and near field communication (NFC) technology. Their algorithm to extract the viewing path is shown in Figure 3 [1, 4].

The algorithm to extract the viewing path [4]
1: **Procedure** extractViewingPath 2: **for** each *Area[i]* in the museum, *i*= 1,2,…,*Area.Count* **do** 3: **if** *User.Area[i].ViewingTime* > *Area[i].avgViewingTime* **then** 4: *User.ViewingPath = Area[i]* 5: **end if** 6: **end for** 7: return *User.ViewingPath*

Figure 3: The algorithm to extract the viewing path [1, 4]

Seo and Ahn [5] also presented a novel method for improving content recommendation accuracy using LBS-based users viewing path similarity. They used the algorithm shown in Figure 4 to estimate the similarity between two paths [1, 5].

```
simPath(User,preUser)
{
int No = intersection(User.Path, preUser.Path);
int TotalNo = User.Path.Count;
pathSim = No/TotalNo;
return pathSim
}
```

Figure 4: The algorithm to estimate the similarity between two viewing paths [1, 5].

In order to enjoy location-based services, users have to disclose their location, People want to use location-based services while they want preserve their location privacy. The authors of [6] proposed a solution for this scenario. The authors of [7] proposed an energy-efficient source location privacy preserving solution. While the authors of [8] proposed a novel trusted third party-free location privacy preserving method.

One of the key techniques needed in LBS development is positioning. For the outdoor positioning, the global positioning system provides general solution. However, indoor positioning is still open and many techniques for indoor positioning have been proposed. Among the proposed methods, the fingerprinting method was investigated by the authors of [9] and [10]. The authors of [9] proposed a new way to

determine the nearest neighbor. The method assigns heavier weights to strong signals. Whereas, the authors of [10] introduced a method to efficiently collect data.

There are so many LBS applications. Vehicle navigation systems, taxi management systems, bus information systems, and human resource management systems are examples of LBSs. The authors of [11] proposed a system for sensing the stray of any objects within user-specified region.

Design of the LBS System

Considering the functions provided by existing LBSs, it was determined that the proposed system should provide the following functions.

- live broadcasts; requires developing a video player
- video on demand
- electronic program guides
- point of interesting (POI) information pushed to users
- display maps
- zoom in on and move maps
- allow users to search for POIs on maps
- allow users to order and purchase souvenirs and local products
- allow users to monitor delivery of a product
- allow users to upload news content
- display information on cultural events
- allow users to purchase coupons
- allow users to select a category of POIs and display the POIs in that category
- allow users to register content with metadata
- provide a bulletin board
- provide user management
- receive and display information pushed by the server
- manage app versions
- be compatible with other system components
- register new content for the LBS
- content location information complying with general code system
- categorized content
- a flexible category system
- story metadata in a back-end system
- transcoding done whenever content is registered
- the ability to use content stored in the back-end system
- IDs of LBS content that complies with the ID system of the back-end system
- after inspection, the LBS registers user created contents in the back-end system
- allow metadata to be updated and content to be deleted
- be compatible with existing user management components
- push messages to many mobile units
- push messages to all mobile units in a certain area
- push messages to all users of a certain age
- push messages to users on their birthday or wedding anniversary

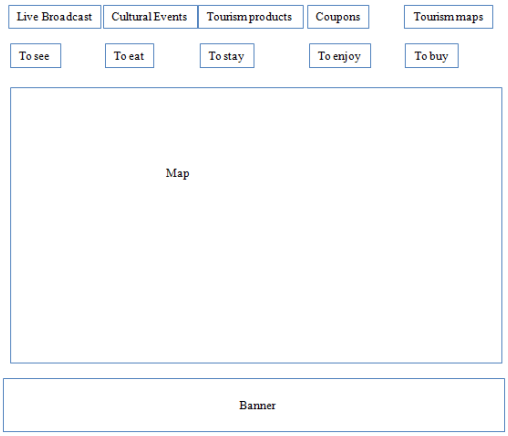

Figure 5: Our design of the main page of the LBS [1]

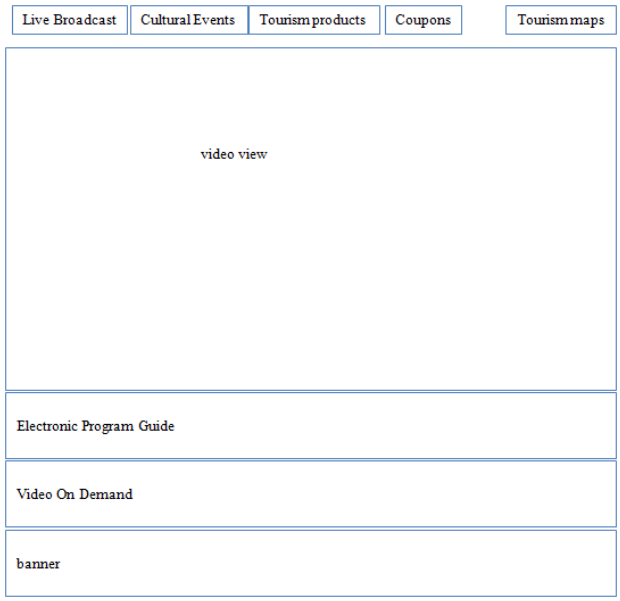

Figure 6: Our design of the User Interface for Live Broadcast

Considering the functional requirements for the LBS system, the main page of the user interface was designed as shown in Figure 5. Points of interest are classified into five categories. For example, if a user clicks the "to see" button, then the museums, exhibition centers and sceneries are marked on the map. If a user clicks a mark, then detailed information of the place associated with the mark is displayed.

If the 'live broadcast' button is clicked, then the user moves to the page shown in Figure 6. The page consists of a video view, video on demand, and an electronic program guide.

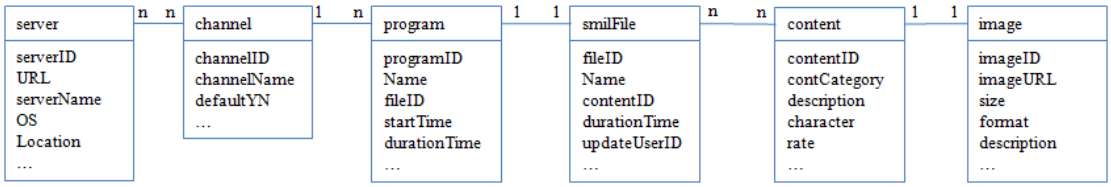

Figure 7: Part of the database for the live broadcast service.

A database was designed for the system. A part of the database for live broadcast services is shown in Figure 7. In the system, there is one-to-one mapping between streaming servers and channels. However, a streaming server may serve multiple channels, and a channel can be served by multiple streaming servers. One of the channels is designated as the default channel, and this channel plays in the video view when the "live broadcast" button is clicked. In the "Electronic Program Guide" box, the user finds a list of channel names that the system provides. Each of the channels in the list is associated with a few thumbnails representing content to be broadcast. The first thumbnail represents the content currently aired on the channel. Other thumbnails represent feature content to be aired on the channel. The system retrieves these thumbnails from the "image" table. Each image is associated with only one piece of content. A content item can be a part of many programs, and a program consists of many content items. A program is represented in a synchronized multimedia integration language (SMIL) file.

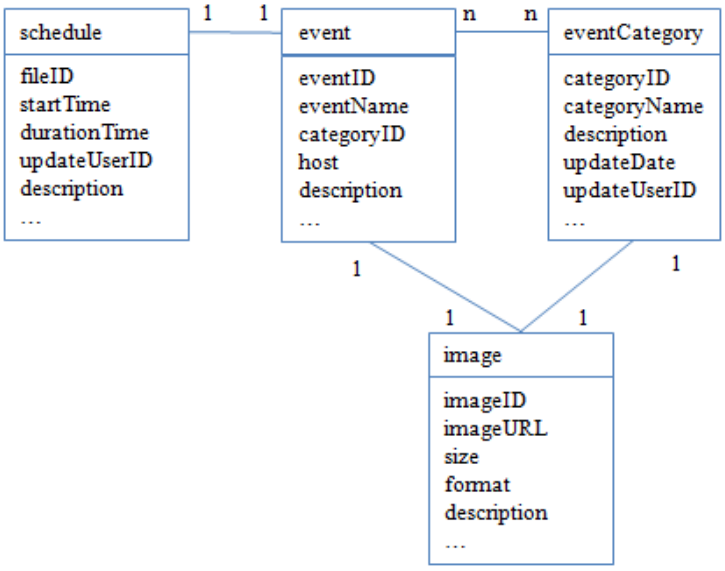

Figure 8: Part of the database to handle the "Cultural Events" menu.

The part of the database needed to handle the "Cultural Events" menu is shown in Figure 8. An event can belong to multiple categories. For example, the opera "Napoleon" can be categorized into opera, music, romance, and so on.

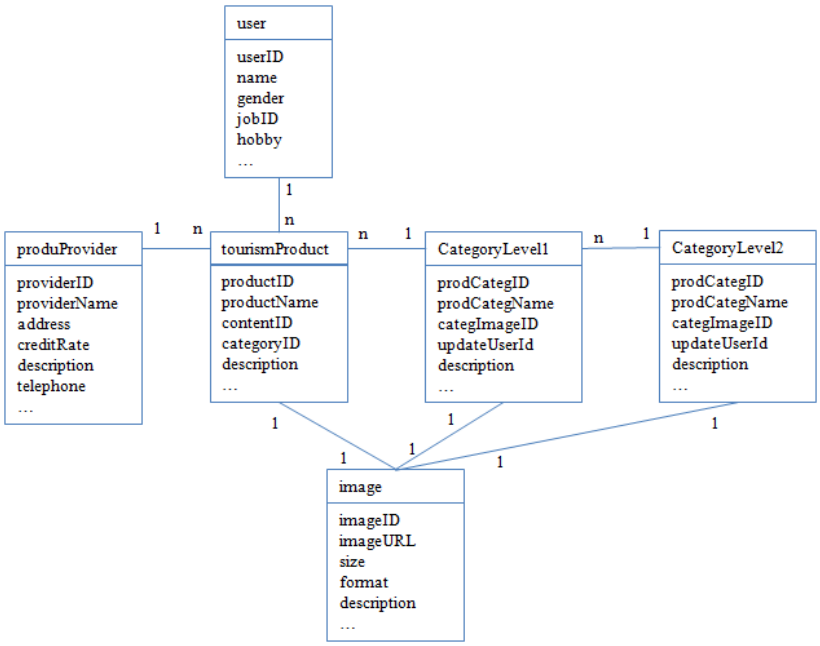

Figure 9: Part of the database to handle the "Tourism Product" menu.

The process of handling the "Tourism Product" menu is complicated. From this menu, users can purchase a product item. Part of the database needed to handle the "Tourism Product" menu is shown in Figure 9. Each product is associated with an image that represents the product. A product provider can supply multiple kinds of products. A person can purchase many kinds of products. Products are categorized. If clothing is one of the categories in level 2, then shirts, trousers, socks, and so on are examples of level 1 categories belonging to clothes.

The process of handling the "Coupons" menu is similar to the process of handling the "Tourism Product" menu, so the explanation for it has been omitted. Part of the database needed to handle "Tourism Maps" is shown in Figure 10. Assuming that users allow the system to collect location information, the user's location information is recorded in the "device" table. In this table, the system can find the user's recent location information. Location information in this table is periodically appended to the file identified by "fileID" for archiving and removed from this table. A person usually carries one smartphone, but may have more than one smartphone. Referring to these tables, the system recognizes the location of the user and finds information about points of interest near the user. The system pushes information on the POIs to the user's smartphone. Part of the database needed to handle the "Tourism Maps" menu is shown in Figure 10.

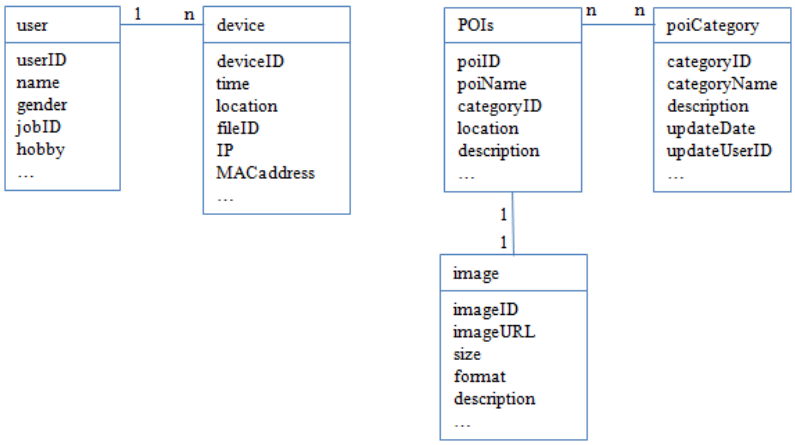

Figure 10: Part of the database needed to handle the "Tourism Maps" menu.

Implementation considerations

Part of the LBS system was implemented. This system is a kind of a client/server system. One of the most important elements of the user interface of the client is the map. For the map, the system uses the Naver Map library, nmaps.jar. After inserting the library in the program, the following sentence was inserted into the AndroidManifest file in order to allow the program to access the network:

<uses-permission android:name="android.permission.INTERNET"/>

Then, the NMapViewer class was defined, extending NMapActivity. This class creates an NMapView, as shown in Figure 11. Then, the class initializes the map view, registers the listeners, obtains controllers, and so on.

```
public class NMapViewer extends NMapActivity {
...
mMapView = (NMapView) findViewById(R.id.mapView);
...
mMapView.setClickable(true);
...
mMapView.setOnMapStateChangeListener(onMapViewStateChangeListener);
...
```

Figure 11: Part of the map viewer definition.

This system allows users to subscribe to the system. In the subscription process, the user has to fill out the form, providing his/her name, gender, birth date, hobby, occupation, and so on. A part of the program to obtain gender information is shown in Figure 12. The spinner object is used in the program.

```
SpinnerGender = (Spinner) findViewById(R.id.spin_sex);
SpinnerGender.setPrompt("Select your gender!");
...
SpinnerGender.setAdapter(adpt_arrCategories);

SpinnerGender.setOnItemSelectedListener(new OnItemSelectedListener() {
public void onItemSelected(AdapterView<?> parent, ..., int position, long id) {
category = (String) adpt_arrCategories.getItem(position);
isSelected = true;
}
public void onNothingSelected(AdapterView<?> arg0) {
isSelected = false;
...
```

Figure 12: Part of the program to obtain the user's gender.

In order to obtain the user's birth date, the system uses DatePicekrDialog, as shown in Figure 13.

```
int DATE_DIALOG_ID = 0;
private void updateDate(){
edit_birth.setText(new StringBuilder() .append(...) ....append(mDay) .append(" "));
}
private DatePicker...Listener mDateSetListener = new DatePickerDialog...(){
@Override
public void onDateSet(DatePicker view, ..., int dayOfMonth) {
mYear=year;
...
updateDate();
...
@Override
protected Dialog onCreateDialog(int id){
switch(id){
case DATE_DIALOG_ID:
return new DatePickerDialog(this, ..., mYear, mMonth, mDay);
}
return null;
}
```

Figure 13: Part of the program to obtain a user's birth date.

After filling out the form, a user can click the "Subscribe" button in order to send the subscription request to the server. The process of invoking the thread that sends the request to the server is defined in the button's "click listener," as shown in Figure 14.

```
button_joinuser.setOnClickListener(new OnClickListener() {
public void onClick(View v) {
dialog = ProgressDialog.show(UserJoinActivity.this, null, "progressing...");
id = edit_id.getText().toString();
name = edit_name.getText().toString();
birthDate = edit_birth.getText().toString();
...
address = edit_address.getText().toString();
pairs = new ArrayList<NameValuePair>();
pairs.add(new BasicNameValuePair("id", id));
pairs.add(new BasicNameValuePair("BirthDate", birthDate));
try {
pairs.add(new BasicNameValuePair("name", ...(name, "UTF-8").toString()));
...
} catch (UnsupportedEncodingException e) {
e.printStackTrace();
}
pairs.add(new BasicNameValuePair("password", password));
Thread thread = new JoinThread();
thread.start();
...
```

Figure 14: Part of the process to invoke the subscription thread.

The server has to be able to add a new user in order to handle a subscription request received from a client. Adding a new user requires access to the database. The UserDao class was defined for accessing the database. The part of the program that adds a new user using the joinUser method defined in the UserDao class is shown in Figure 15.

```
String id = request.getParameter("id");
String birthDate = request.getParameter("BirthDate");
String password = request.getParameter("password");
String name = URLDecoder.decode(request.getParameter("name"),"UTF-8");
...
result = userdao.joinUser(id, name, birthDate, sex, password, address);
if(result == 1){     //success
out.print("join ok");
}
else if(result == 0)
...
```

Figure 15: The part of the server program to add a new user.

A client collects global positioning system (GPS) data whenever its location changes, and it sends the current location and time to the server whenever the user stops and does not move. Part of the process to collect GPS data is shown in Figure 16. The GPS receiver installed in a smartphone is extremely sensitive, and the onLocationChanged method is executed even when the smartphone is slightly shaken by the user.

```
public void onLocationChanged(Location location) {
...
_userInfo.setDate(newDate(System.currentTimeMillis()));
_userInfo.setLatitude(location.getLatitude());
_userInfo.setLongitude(location.getLongitude());
gps_current_speed = location.getSpeed();
gps_accuracy = location.getAccuracy();
gps_altitude = location.getAltitude();
...
```

Figure 16. Part of the process to collect GPS data.

A client sends its current location to the server when the client does not move. The part of the process that sends the current location is shown in Figure 17. The system concludes that the client is not moving when the speed attribute of the GPS data is zero. Once the client sends the current location, it does not send GPS data to the server until the user's next stop. In order to prevent sending the same location repeatedly while the user is not moving, the system uses the 'send' variable in the process.

```
private class UIChangingThread extends Thread {
public void run() {
while (true) {
...
if(gps_current_speed==0 && send==false){
_userInfo.requestToServer();
...
send = true;
}
else if(gps_current_speed!=0){
...
send=false;
}
...
try {
UIChangingThread.sleep(1000);
} catch
...
```

Figure 17. Part of the process sending GPS data to the server

Conclusions

A tourist guide application must be aware of the user's current location and must provide services based on the location. After reviewing recently published location-based services, this paper introduced the design of a location-based service system for local area tourism. The system is a client–server system. Implementation considerations for both client and server were also discussed. Further research will include developing a practical system.

Acknowledgment

This work was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (NRF-2011-0006942) and by 'Development of Global Culture and Tourism IPTV Broadcasting Station' Project through the Industrial Infrastructure Program for Fundamental Technologies funded by the Ministry of Knowledge Economy (10037393).

This paper is a revised and expanded version of a paper entitled "Review of the Techniques for Location Based Service Development" presented at The 2015 International Interdisciplinary Workshop Series, Jeju, Korea, April 15-18, 2015.

References

[1] Yim, J. 2015, "Review of the Techniques for Location Based Service Development. Advanced Science and Technology Letters 86 (ASTL 86), pp.68-71.
[2] Chen, P., Gu, J., Zhu, D., and Shao, F., 2013, "A Dynamic Time Warping based Algorithm for Trajectory Matching in LBS," IJDTA 6(3), pp.39-48.
[3] Chen, P., Gu, J., Lin, X., and Tan, R., 2012, "A Probabilistic Approach for GNN Queries in LBS," IJMUE 7(2), pp.189-194.
[4] Seo, Y., and Ahn, J., 2013, "Hybrid Contents Recommendation Service Using LBS and NFC Tagging," IJSH 7(5), pp.251-262.
[5] Seo, Y., and Ahn, J., 2013, "Novel Method for Enhancing Contents Recommendation Accuracy Using LBS-based Users Viewing Path Similarity," IJMUE 8(4), pp. 217-227.
[6] Ashouri-Talouki, M., and Baraani-Dastjerdi, A., 2012, "Homomorphic Encryption to Preserve Location Privacy," IJSIA 6(4), pp.183-190.
[7] Song, S., Park, H., and Choi, B., 2013, "E-LPG: Energy Efficient Location Privacy Scheme Against Global Attackers in Sensor Networks," IJSIA 7(2), pp.27-46.
[8] Yang, N., Cao, Y., Liu, Q., and Zheng, J., 2014, "A Novel Personalized TTP-free Location Privacy Preserving Method," IJSIA 8(2), pp.387-398.
[9] So, J., Lee, J., Yoon, C., and Park, H., 2013. "An Improved Location Estimation Method for Wifi Fingerprint-based Indoor Localization," IJSEIA 7(3), pp.77-86.
[10] Park, G., Jeon, M., and Oh, C., 2014, "Indoor Wireless Localization Using Kalman Filtering in Fingerprinting-based Location Estimation System," IJSEIA 8(1), pp.235-246.
[11] Jeon, B., and Kim, Y., 2013, "A System for detecting the Stray of Objects within User-defined Region using Location-Based Services," IJSEIA 7(5), pp.355-362.