Implementation of Building Recognition Android App[1]

Jaegeol Yim

Dongguk University at Gyeongju Korea
yim@dongguk.ac.kr

Abstract

Augmented Reality (AR) is a view of a real-world mixed with and/or superimposed by virtual objects. Therefore, one of the most important key techniques of implementing an AR is recognizing the environment so that it can correctly determine the virtual objects to be displayed on the multimedia representing the real-world. This paper reviews the "Method of Recognizing Objects in a Picture for Mobile AR Using Electronic Maps." Then we describe the implementation of the method and an application on an Android mobile in detail.

Keywords: *Augmented Reality, Mobile App, Object Recognition*

1. Introduction

Augmented Reality (AR) is a variation of Virtual Reality (VR). VR is defined as "the use of real-time digital computers and other special hardware and software to generate a simulation of an alternate world or environment, which is believed as real or true by the users" [1]. That is, VR creates an environment, and the user feels that he/she is in that environment. On the contrary, in AR, the user watches an image of the real world which is more vivid or more informative because of the multimedia contents additionally displayed on the image.

By the definition of AR, the first thing an AR has to do is recognizing the real surrounding world in the image so that it can correctly select the multimedia contents to be additionally displayed on the image of the real world. Most the methods of recognizing the image compare the image with the images in the database. But, [2] does it differently. Making use of sensor values it recognizes the object in the picture and displays additional content. This paper describes the implementation of the application in detail.

As the electronic techniques advance, computing machines have been miniaturized and smart phones are equipped with powerful processors and large memories. Consequently, various services have become available on smart phones. Since a smart phone is a personal device, it is an excellent candidate device on which a context-aware service may be provided. As an example of context-aware service on smart phones, they have picked the campus guide and introduced their implementation of it in [2].

The campus guide consists of a server and client. The main features of the client include determining the current location of the user and the building the user is watching, and playing the video that is closely related to the building.

In order to realize the client's features, the server consists of many components including the streaming server and a database server. The streaming server takes charge of delivering video content to the client, whereas the database server takes charge of

[1] An earlier version of this paper was published as "Development of a Prototype of Campus Guide Mobile Application"

storing and retrieving information of the videos. When a new video is obtained and stored in the archive, the path to the video file will be stored in the database.

The service scenario of the campus guide is summarized as follows:

1) When the application is first downloaded by a user, the application takes the user's personal information and saves it in the database.

2) When the user starts running the application, it renders the campus map.

3) When the user takes a picture of a building, the application recognizes the building.

4) The application displays the picture along with a text showing some information about the picture.

5) The application plays the video closely related to the building.

6) After playing a video, go back to step 2.

This paper is focusing on Step 3 of the above process and describes the implementation of the building recognition Android app in detail. The remaining part of this paper organized as follows: Building recognition is a vital component of the campus guide, and the campus guide is a kind of context-aware AR VOD system. Therefore, VOD, context-aware, and AR are this paper's related topics. These topics will be discussed in section 2. Section 3 and 4 introduce our design and implementation of the app. Our experimental results are discussed in Section 5 and our concluding remarks will be shown in Section 6.

2. Related Works

The campus guide is a kind of VOD (video on demand) system because it plays a video when the user selects the menu. It is a mobile VOD service since it runs on a smart phone. There are so many published studies regarding mobile VOD. In [3], cache schemes were proposed to reduce the waiting time of VOD clients. Ensuring service continuity between fixed and wireless networks has been one of hot research topics. In [4], an approach was proposed to appropriately select a broadcasting scheme that minimizes delays. In [5], an apparatus was proposed for easily setting up an IPTV interactive digital channel.

Context-aware service is desired, but there are not many commercially successful context-aware services. [6] focuses on the methodology of context-aware service design. It starts with building a Petri net model of the system and refines the model into a system design. Context-aware services heavily rely on specific context information and cannot be realized in highly distributed and decentralized environments. [7] addresses this problem by combining the ideas of context-aware frameworks and autonomous data dissemination.

The campus guide [2] is a kind of context-aware VOD and also an AR because it is aware of the user's location and situation of taking a picture, plays a video, and displays multimedia contents together with the picture taken by the user. The video and multimedia contents should be something that is related to the object in the picture. That implies that recognizing the object in the picture is extremely important. We describe the implementation of building recognition part of it in detail.

3. Design

Whenever the user takes a picture, the application recognizes the building in the picture taken and plays the video related to the building. The application repeats this process until the user terminates the application. This process is summarized in Figure 1.

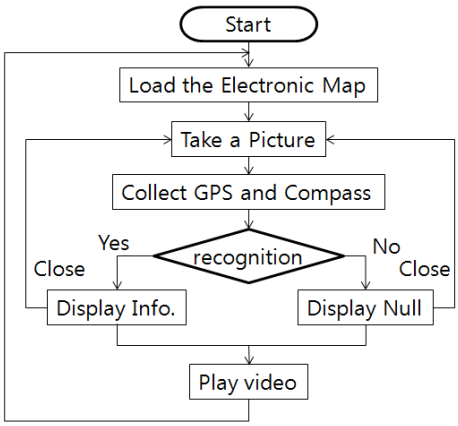


Figure 1. The Process of the Application

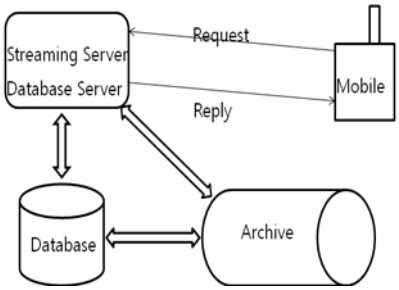


Figure 2. Main Components of our System

The application plays the video that is related to the building in the picture. Video files are usually huge and stored in an archive system. Information about videos in the archive should be stored in a database for easy retrieval. The database has many tables including the following three: 1) genreTable consisting of genreID, genreName, genreIcon, and description attributes; 2) videoTable consisting of videoId, videoTitle, videoPath, metadata, …, attributes; and 3) genreVideoTable consisting of genreID and videoId attributes. The type of metadata attribute in videoTable is defined by XML, and the video metadata file will be saved here.

Since the video files are stored in an archive, the system should be a kind of client-server system as shown in Figure 2. On the server, in addition to a database server, a streaming server that takes charge of controlling and delivering media is needed. There are quite a few free streaming servers, and implementing a streaming server is quite easy. They have installed Darwin [8] open source streaming server as their streaming server. It is run on Microsoft Windows Server 2008.

The method of recognizing objects in photos captured via a smart phone camera used in the application utilizes electronic maps. So far, the accessories attached to a mobile phone have not been accurate enough to locate the phone on a map, determine the

camera orientation, or measure the focus distance of the camera. Therefore, all existing picture recognition systems rely on image processing techniques. Many of them do make use of GPS and compass data to narrow down the scope of the images to be compared with the photo image, but they all rely on image processing techniques at the final stage of photo recognition. Now, it is changed. They have been improved, and the measurement error of a GPS (compass) on a recently released smart phone is about 10 meters (less than 2 degrees). The process of recognizing the building taken by the camera on the smart phone is described as follows:

1) A user obtains an electronic map describing the physical area of the application system. Considering an application of museum guide, the user need a drawing (AutoCAD drawing, for example) of the museum building. There exist drawings for all large buildings. The level of detail of the drawing is closely related to the purpose of the application. Considering a campus guide on the building level, the object recognition process should be able to identify the name of the building on the photo. An example of an electronic map for this application is shown in Figure 3. The electronic map consists of edges representing the outline of the Natural Science Building, Gymnasium Building, Student Hall, and so on, where an edge is represented by a pair of points, its start and end. A point is represented by a pair of real numbers, (longitude, latitude).

```
NaturalScience
129.198050, 35.862546
...
129.198050, 35.862546
Gymnasium
129.196986, 35.862514
...
129.196986, 35.862514
StudentsHall
129.195878, 35.862025
...
129.195878, 35.862025
...
```

Figure 3. A Part of an Electronic Map of a University Campus

2) With the sensor data, the application determines the location of the smart phone. In Android, “LocationManager” class provides a method that returns the location of the smart phone. Most smart phones are equipped with a GPS receiver, WIFI device, 3G or 4G communication device. “LocationManager” uses data from these devices to determine the location of the smart phone.

3) The application collects the orientation data, namely azimuth, pitch and roll values. In Android, “SensorManager” class provides a method that returns those values.

4) Using the “Camera” class, the application obtains the focus distance of the camera. In Android, “Parameters” class nested in “Camera” class provides a method, getFocusDistance, which returns the focus distance.

5) The application executes “Object Recognition Algorithm” to identify the object on the camera. This algorithm calculates the formula representing the line of sight of the camera with the location and orientation data, finds the element (edge) of the electronic map intersecting with the line of sight, and its distance is close to the focus distance. If

there is no such element, then the algorithm concludes that the object on the photo is not something represented on the electronic map, maybe a person. If there is such an edge (element), then the building (the point of interest) whose outline contains the edge, is determined as the object in the photo. Our "Object Recognition Algorithm" is described in Figure 4.

objectRecognitionAlgorithm (fd, location, azimuth, pitch, eMap)

// fd: focus distance ;

// location: (longitude, latitude, altitude)

Step 1: With location, (x, y, z), and azimuth, calculate the following formula whose slope is obtained from the azimuth value.

$ax + b = y$ ---- (eq. 1)

Step 2: Find S = { s | s is an edge in eMap and s intersects eq. 1}.

Step 3: If S is an empty set then return NIL

Step 4: Find the edge e in S which is closest to location. Delete e from S. Let the building whose outline contains e be a "candidate building" and the intersect point of e and eq. 1 be (x', y').

Step 5: Let the distance between (x, y) and (x', y') be d.

Step 6: If "z+d*tan(pitch)" is between the bottom and the top of the "candidate building" then jump to Step 8

Step 7: Go to Step 3

Step 8: Let the distance between (x, y, z) and (x', y', x+d*tan(pitch)) be dist. If (|dist – fd| < threshold) then return "candidate building" else return NIL, where threshold is a small number representing the error of getFocusDistance.

Figure 4. The Algorithm of Recognizing the Building taken by the Camera

4. Implementation

They implemented the client on Android [9]. For the purpose of rendering a video stream, MediaPlayer and VideoView are usually used. They chose to use VideoView in their ***MyMediaPlayer*** because it is easier to handle. What they need to do is to override just one method, onCreate, of Activity as shown in Figure 5. VideoView plays a video streamed by setVideoURI(). If we want to play a local video, then we can use setPath() instead of setVideoURI(). ***Path*** in the code contains video's URL.

```
setContentView(R.layout.videoviewplayer);
videoView = (VideoView) findViewById(R.id.videoView);
videoView.setVideoURI(Uri.parse(path));
MediaController mediaController = new MediaController(this);
videoView.setMediaController(mediaController);
videoView.requestFocus();
videoView.start();
```

Figure 5. A Part of the Player Module

In the previous section, we said that the main purpose of the database is to store and retrieve video information. In addition to the main purpose, they plan to use the database to provide context-aware services. To this end, they have tables of recording subscribers' information, who watched what video, to what genre the video belongs to, and so on, as is shown in Figure 6. Some of the tables are further explained in the following.

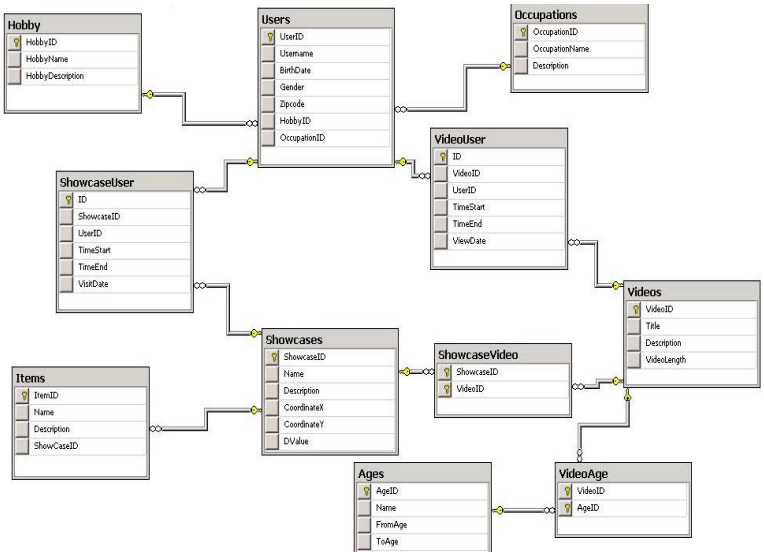


Figure 6. A Description of the Database

GenreTable is to save a list of gene names. The structure of the table is shown in Figure 7. Instances of GenreName include comedy, drama, animation…etc.

No.	*Field Name*	*Data type*	*Notes*
1	*GenreID*	*Int*	*Primary Key, Identity (1,1)*
2	*GenreName*	*nvarchar(50)*	
3	*IconPath*	*nvarchar(150)*	*Path to image represented for this genre*
4	*Detail*	*nvarchar(200)*	*Short describe for this genre*

Figure 7. The Structure of GenreTable

VideoTable is to save a list of video names. The structure of the table is shown in Figure 8. Instances of VideoTitle include "Romeo and Juliet", "Lion King", "Alexander the Great", …, etc.

No.	*Field Name*	*Data type*	*Notes*
1	*VideoID*	*Int*	*Primary Key, Identity (1,1)*
2	*VideoTitle*	*nvarchar(50)*	
3	*VideoPath*	*nvarchar(150)*	*URL of this video*
4	*IconPath*	*nvarchar(150)*	*Path to image describe for this video*
5	*Detail*	*nvarchar(200)*	*Short describe for this video*
6	*VideoLength*	*Int*	*Length of this video*

Figure 8. The Structure of VideoTable

GenreVideo table saves information of "which video belongs to what genre." The structure of the table is shown in Figure 9. GenreID and VideoID are foreign keys from GenreTable and VideoTable.

No.	*Field Name*	*Data type*	*Notes*
1	*GenreID*	*Int*	*Primary Key, Foreign Key*
2	*VideoID*	*Int*	*Primary Key, Foreign Key*

Figure 9. The Structure of GenreVideo Table

The application plays a video after recognizing the building in the picture. The video played is the one mostly related to the building. In the future, the application can be extended to provide VOD service. The service will be aware of the user's age, hobby, occupation and other personal information and recommend videos for the user. AgesTable is to save a list of all age types. The structure of AgesTable is shown in Figure 10.

No.	*Field Name*	*Data type*	*Notes*
1	*AgeID*	*Int*	*Primary Key, Identity (1,1)*
2	*AgeName*	*nvarchar(50)*	
3	*FromAge*	*Int*	*This type of this age has age from FromAge*
4	*ToAge*	*Int*	*This type of this age has age to ToAge*

Figure 10. The Structure of AgesTable

Similarly to AgesTable, HobbiesTable saves a list of all hobby names as shown in Figure 11.

No.	*Field Name*	*Data type*	*Notes*
1	*HobbyID*	*Int*	*Primary Key, Identity (1,1)*
2	*HobbyName*	*nvarchar(50)*	
3	*HobbyDescription*	*nvarchar(150)*	*Short description for this hobby*

Figure 11. The Structure of HobbyTable

Similarly to AgesTable, OccupationsTable saves a list of all occupation names as shown in Figure 12.

No.	*Field Name*	*Data type*	*Notes*
1	*OccupationID*	*Int*	*Primary Key, Identity (1,1)*
2	*OccupationName*	*nvarchar(50)*	
3	*Description*	*nvarchar(150)*	*Short description for this occupation*

Figure 12. The Structure of OccupationsTable

UsersTable is to save information of the subscribers' personal information as is shown in Figure 13.

No.	Field Name	Data type	Notes
1	UserID	Int	Primary Key, Identity (1,1)
2	Username	nvarchar(50)	
3	BirthDate	Date	
4	Gender	Bit	
5	Zipcode	nvarchar(20)	
6	HobbyID	Int	Foreign Key
7	OccupationID	Int	Foreign Key

Figure 13. The structure of UsersTable

VideoAge is a table to save information of "which video is for what ages" as is shown in Figure 14. A tuple of the table implies that the video represented by the VideoID is for the users of the age represented by the AgeID.

No.	Field Name	Data type	Notes
1	VideoID	Int	Primary Key, Foreign
2	AgeID	Int	Primary Key, Foreign

Figure 14. The Structure of VideoAge

VideoUsers table is to save information of "who watched which video" as is shown in Figure 15. A tuple of the table implies that the person represented by UseID watched the video represented by VideoID from TimeStart to TimeEnd on ViewDate.

No.	Field Name	Data type	Notes
1	VideoID	Int	Primary Key, Foreign
2	UserID	Int	Primary Key, Foreign
3	TimeStart	Time	
4	TimeEnd	Time	
5	ViewDate	Date	The date when this user saw this video

Figure 15. The Structure of VideoUsers

After watching a video, a user is supposed to evaluate the video. RatingTable is to save users' evaluation as is shown in Figure 16.

No.	Field Name	Data type	Notes
1	VideoID	Int	Primary Key, Foreign
2	UserID	Int	Primary Key, Foreign
3	Rating	Float	
4	Time	Datetime	The time when this user evaluated this video

Figure 16. The Structure of RatingTable

The mobile application consists of the classes of IdentifySubjectMain, SHCameraSurface, aPoint, MathMethod, and so on as shown in Figure 17.

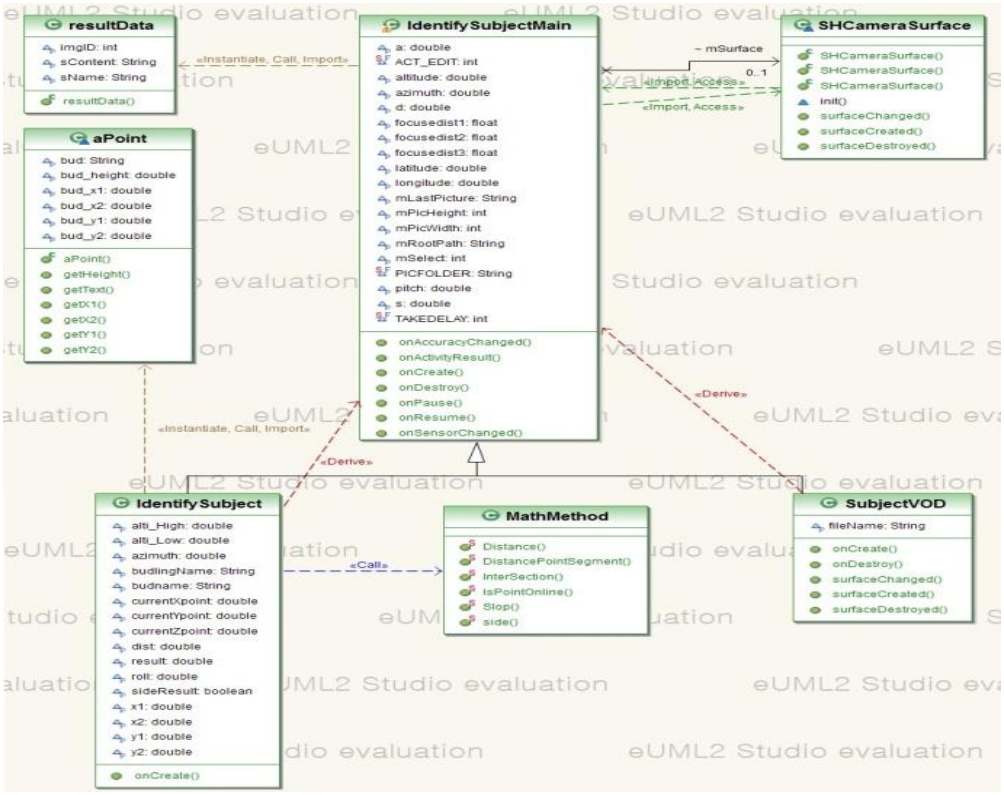


Figure 17. Class Diagram of the Application

IdentifySubjectMain contains methods of obtaining GPS information to determine the user's location, of scanning sensors to collect sensor values, of taking a picture with the camera, and so on, as shown in Figure 18. Using the LocationManager class we can easily obtain GPS information. In order to instantiate this class, we use the following sentence:

Context.getSystemService(Context.LOCATION_SERVICE);

Then, we register the listener (mListener) with the Location Manager (mLocMan) to receive location updates:

mLocMan.requestLocationUpdates(LocationManager.GPS_PROVIDER, 1000, 0, mListener).

The Location Manager (mLocMan) calls onLocationChanged(Location location) defined in mListener when the user location changes. That means, we have to define onLocationChanged(Location location) in mListener that is a LocationListener().

In order to access the device's sensors, we have to use SensorManager. The following sentence creates a new instance of SensorManager:

getSystemService(SENSOR_SERVICE);

The SensorManager calls onSensorChanged(**int** sensor, **float**[] values) when there is change on the registered sensor. So, we register SENSOR_ORIENTATION as follows:

sm.registerListener(**this**, SensorManager.SENSOR_ORIENTATION);

Among the sensor values, the application uses azimuth and pitch as follows:

```
if (sensor == SensorManager.SENSOR_ORIENTATION) {
        azimuth = values[0]-7.5;
        pitch = values[1];
```

Note how they get the azimuth value; 7.5 is the difference between the magnetic north and the map north.

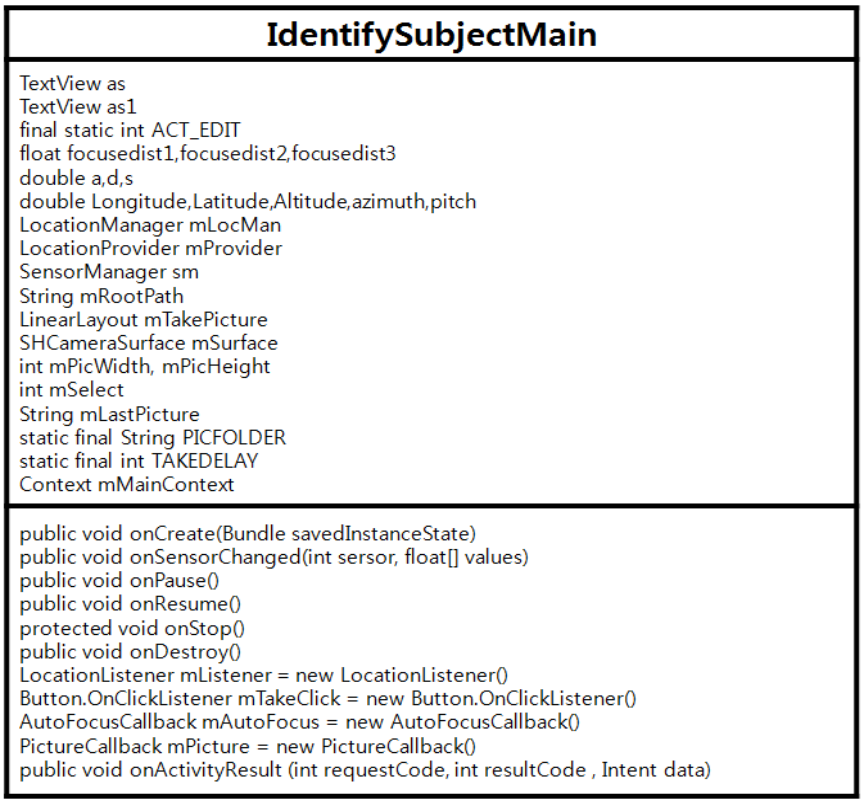


Figure 18. The structure of IdentifySubjectMain

SHCameraSurface contains methods of showing the picture taken by the camera as shown in Figure 19. If SHCameraSurface(Context context) is invoked, init(context) is automatically invoked.

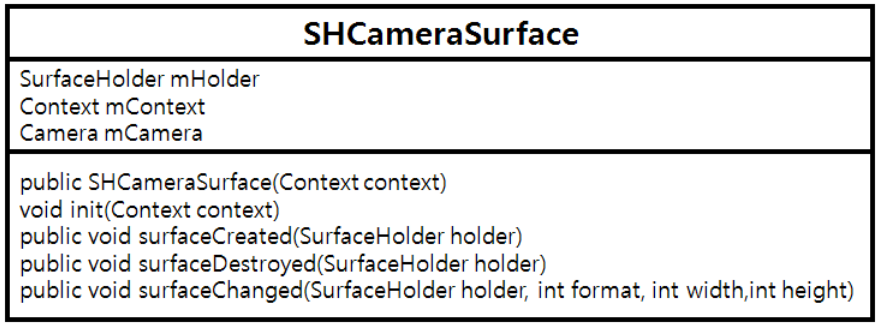


Figure 19. The Structure of SHCameraSurface

The class aPoint is to record the coordinates of the two points defining an edge as shown in Figure 20. It has methods of returning the coordinates of the points.

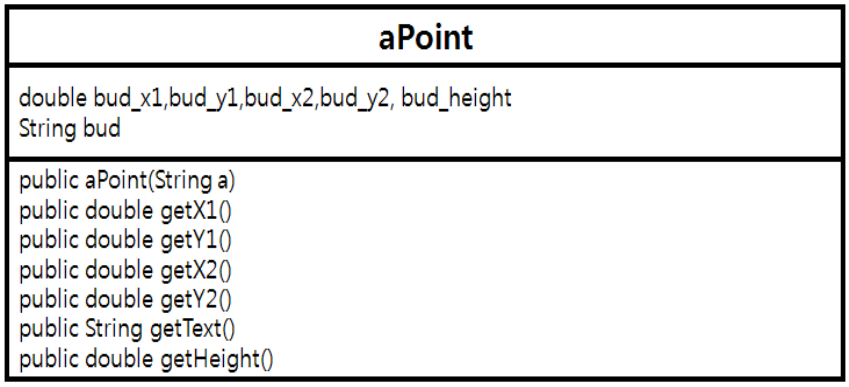


Figure 20. The Structure of aPoint

The most important class of this application is IdentifySubject shown in Figure 21. This class has one method onCreate(). IdentyfySubject is the class of identifying the building in the picture. The process described in Figure 4 is implemented in this class.

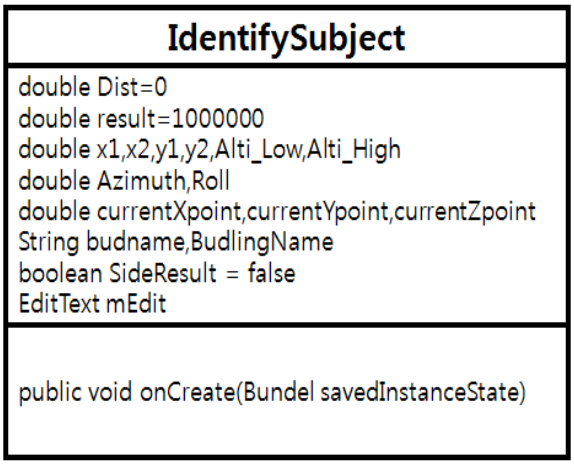


Figure 21. The Structure of IdentifySubject

In the MathMethod class, many mathematical functions are implemented as shown in Figure 22. These functions are mostly used by IdentyfySubject. Given an angle, a point and a line segment, DistancePointSegment() returns the distance from the point to the line if the line defined by the angle and the point intersects the given line segment. Given an angle obtained from the sensor, Slope(angle) returns the slope determined by the angle. Given three points p1, p2, and p3, IsPointOnline() determines if p1 is on the line defined by p2 and p3.

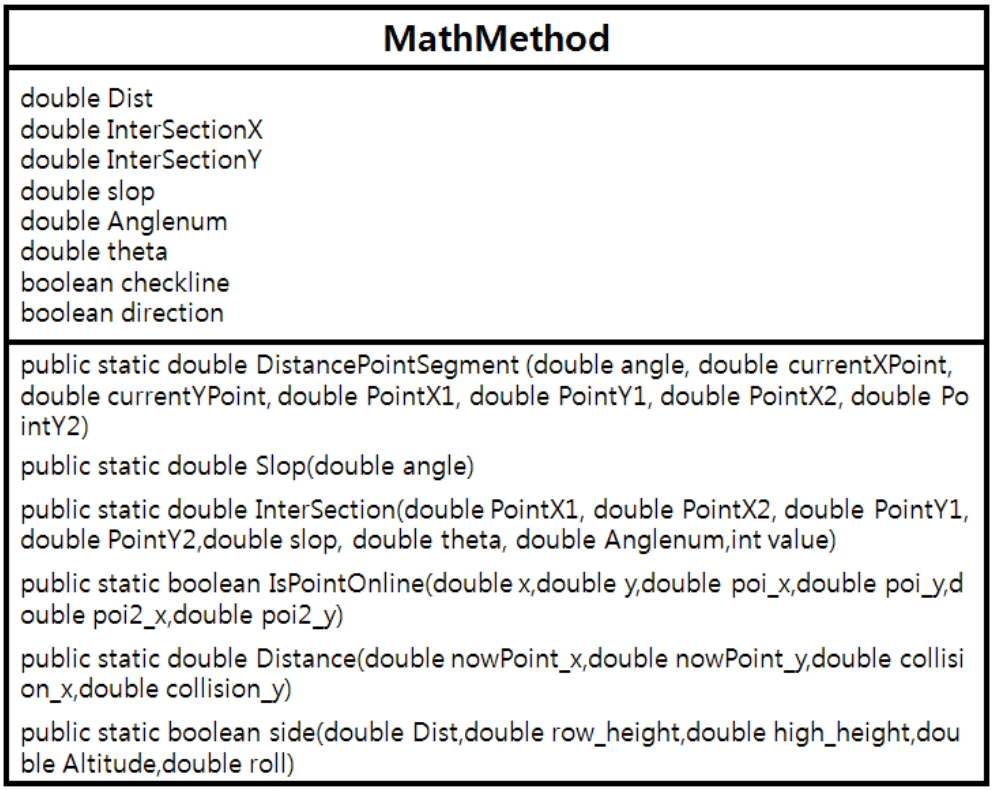


Figure 22. The Structure of MathMethod

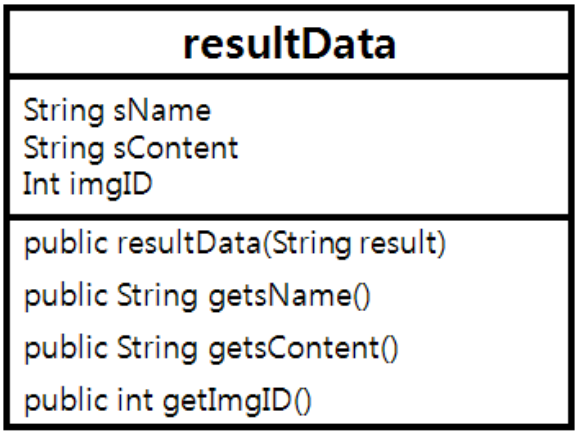


Figure 23. The Structure of resultData

resultData.java is a file where resultData class is defined as shown in Figure 23. resultData crates a popup window and prints out the name of the building.

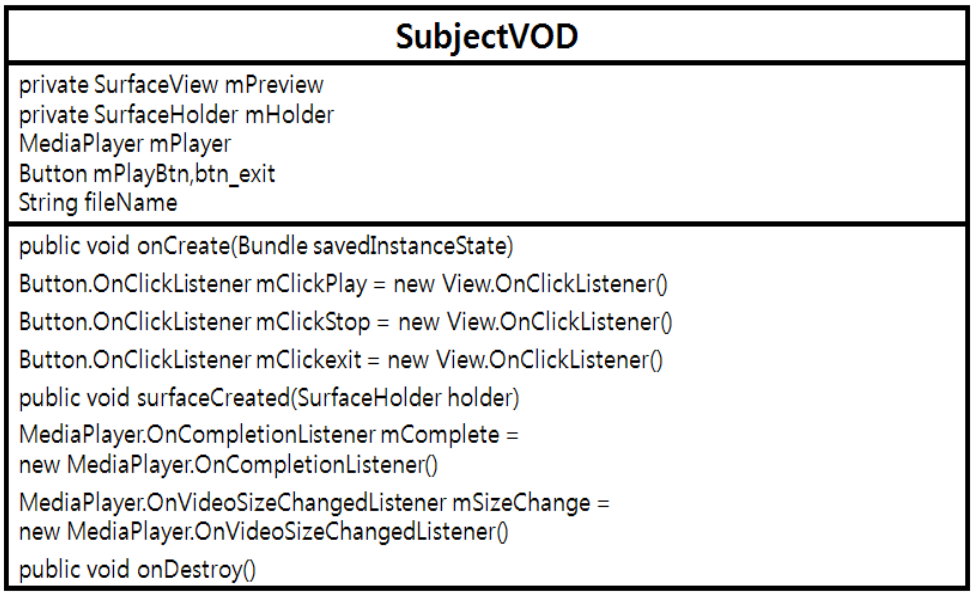


Figure 24. The Structure of SubjectVOD

SubjectVOD is a class to play a video as shown in Figure 24. The user interface of playing a video consists of three buttons (play, stop, exit) and a screen called a surfaceview on which the video is rendered. On each of these three buttons an OnClickListener is defined. surfaceCreated() creates a media player object and defines the path to the video to be played. The names of the other methods are self-explanatory.

5. Experiments

We have performed experiments to test the accuracy of LocationManager at many different locations. An example of the locations is as follows:

A: 129.196990, 35.861779

They obtained the coordinates from Google Map.

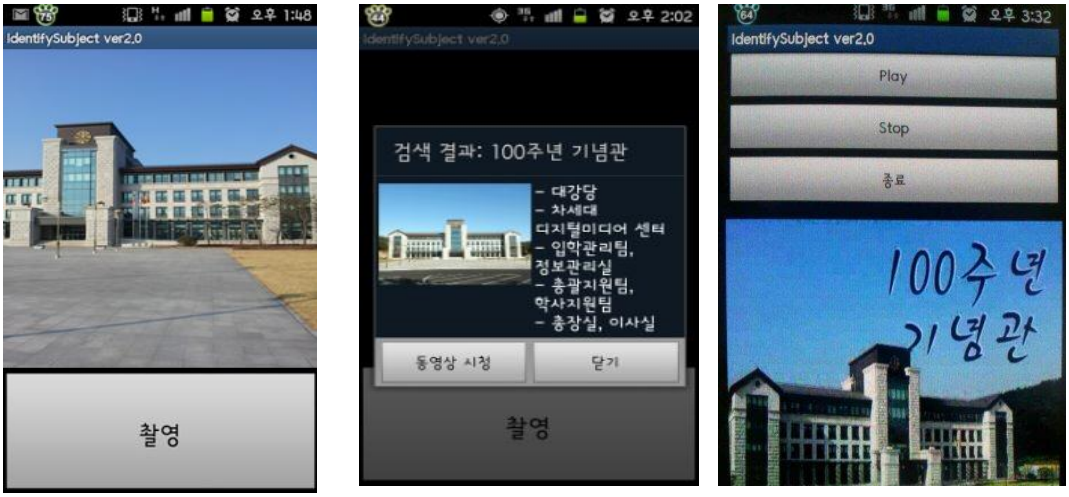
The measurement obtained at A by LocationManager on a Galaxy S2 was (129.1968912, 35.8619749). The distance from longitude 129 degrees to 130 degrees at latitude 36 degrees is about 91.29 m, and the distance from latitude 35 degrees to 36 degrees is about 110.941 m. Therefore, we can conclude that the average error of the measurements obtained at location A is about 13.73948193 m. Considering all the measurements obtained from the other locations, we concluded that the average error of LocationManager on Galaxy S2 phone is 11.52493934 m [2].

After testing LocationManager, we performed experiments to test the accuracy of the compass on a Galaxy S smart phone. In these experiments, we measured the azimuth of our Galaxy S while it is aimed at point C (129.196499, 35.862063) from point A (129.196271, 35.861620). Referring to these coordinates, we can find that the slope of the line defined by those two points is 2.361227042.

The azimuth of the line of sight defined by points A and C can be calculated by the following equation:

$$\text{Azimuth} = 90 - \text{Atan}(\text{Slope})$$

The azimuth we obtained was 22.95307631. This is the azimuth on the map. The magnetic north is not the same as the map north. Using a military compass and map, we found the difference to be about 7.5 degrees. From these calculations and measurements, we concluded that the azimuth of our Galaxy S should be 15.45307631. After many measurements of the azimuths, we concluded that the average error of our measured azimuths is about 0.4 degrees.



(a) taking a picture (b) displaying multimedia (c) playing a video

Figure 25. Example screenshots of the app

6. Conclusions

This paper described our implementation of building recognition part of our campus guide mobile application. Making use of sensor data and electronic maps, the building recognition component identifies the building when the user takes a picture of it. The process of identifying the building in a picture is the first implementation of the idea proposed in [2]. Our experimental results showed that the application recognizes the building taken by the camera practically well.

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(2011-0006942) and by 'Industry, Academy and Research Institute Co-development of Techniques' funded by the Small and Medium Business Administration (R00046281).

References

[1] M. Hincapie, A. Caponio, H. Rios and E. Mendivil, "An introduction to Augmented Reality with applications in aeronautical maintenance", Proceedings of the 13th International Conference on Transparent Optical Networks (ICTON), **(2011)**, pp. 1-4.
[2] J. Joo, J. Yim, G. Lee and T. Le, "Development of a Prototype of Campus Guide Mobile Application", to be published.

[3] P. Sumari and A. Rahiman, "Caching Scheme for Handheld Device in Mobile Video-on-demand System," CGIV, **(2010)**, pp. 49-54.
[4] S. Park, M. Hwang, S. Moon and H. Youn, "An Efficient VoD Scheme Providing Service Continuity for Mobile IPTV in Heterogeneous Networks", CIT 2010, pp. 2589-2595.
[5] M. Lee, "The Service Generation Apparatus for IPTV Interactive Digital Channel", International Journal of Advanced Science and Technology, Vol. 10, September, pp. 37-50 **(2009)**.
[6] T. Lu, Q. Hao, "Scenario-Based Context-Aware Service Design," ICSSSM 2010, pp. 1-6 **(2010)**.
[7] `C. Jacob, D. Linner, I. Radusch, S. Steglich, "Context-aware Data Dissemination and Service Adaptation," ISTMWC 2007, pp. 1-5 9 **(2007)**.
[8] Darwin Streaming Server. Darwin Streaming Server. Available Online, Jan. 2012. http://dss.macosforge.org.
[9] http://developer.android.com.

Authors

Jaegeol Yim

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 Science 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.