http://dx.doi.org/10.5392/IJoC.2013.9.1.018

# GPS-based 3D View Augmented Reality System for Smart Mobile Devices

**Phuc Vo, Chang Yeol Choi***
Department of Computer and Communications Eng.
Kangwon National University, Chuncheon, 200-701, Korea

***ABSTRACT***

*Recently, augmented reality has been proved immensely useful on a day to day basis when tied with location-based technology. In this paper, we present a new method for displaying augmented reality contents on mobile devices. We add 3D models on the view of the camera and use location-based services, motion sensors to calculate the transformation of models. Instead of remaining at a fixed position on camera view while moving around a 3D model, the model rotates on display in the opposite direction that the user is walking. We also design client as a ubiquitous client to reduce constraints on disk space and memory capacity on mobile devices. Implementation results show effective use in creating GPS-based 3D view augmented reality contents for Smart Mobile Devices.*

***Keywords****: Augmented Reality, Location-based, Positioning, 3D Transformation, Mobile Sensors.*

## 1. INTRODUCTION

The futuristic science fiction films with controlled headsets or bionic eyes that enable someone to see live information around them are examples of augmented reality technology. The best way to get a hold of this intriguing technology is to fill smartphone with mobile augmented reality applications. Augmented reality applications are more than just a trendy accessory for mobile phone. They are very useful, educational and of course entertaining. The latest smartphones with high-resolution camera, fast 3D graphics, orientation sensing, GPS and ubiquitous connectivity offer challenges for 3D augmented reality.

Consider the scenario of a tourist visiting an ancient city with a history of illustrious trophies. The tourist shows great interest in real prospect of the city at current standing in various historical periods. In addition, the tourist also wants to walk through or move around an object to admire it thoroughly. Monitoring through video clips and descriptive materials might be difficult. Augmented reality solutions alleviate part of this difficulty, but they still require users to make associations between abstract representations of their environment such as markers and their physical surroundings. This correlation requires significant cognitive effort. The use of location-based services and three-dimensional (3D) geographic information system (GIS) in augmented reality is a natural way for people to explore the real world and can decrease cognitive load.

The number of mobile applications using augmented reality is growing quickly. Augmented reality applications fall into for displaying location-based 2D annotations [1], [6], [14] with camera poses and for displaying marker-based 3D models with object recognition [2]-[5], [13]. Because of missing spatial information, the first category is often difficult to interpret information. Displaying 2D cannot cover complex 3D models. The second one provides information using object recognition and is just available if the user looks to a known object due to location independent services without constraining the individual to a specific geographical location. The scene contains only unknown objects. The system will lose abilities to provide information and no ways help users to recognize their surroundings. Moreover, storing too much data at the client is clearly not well suited because of huge information of the real world. The real world contains diverse types of information requiring space in memory. Limitations in disk space and memory capacity can impose considerable restrictions on mobile applications.

In this work, we present a new method for displaying augmented reality contents on mobile devices. Instead of simply adding annotations on the camera view or using markers to determine the locations of the objects, we add 3D GIS models on the view of the camera and using location-based services, motion sensors to calculate the transformation of models. The application displays automatic rotating 3D model. The 3D information displays based on camera poses computation and self-transforming of models under the changing of geo-positioning and orientation. The 3D model does not remain at a fixed position on camera view while we are moving around its location. This means the 3D model rotates on display in the opposite direction where the user is walking. Then, the 3D model fixes on the ground and gives a more realistic augmented reality.

To solve problems of limitations in disk space and memory capacity, we also implement client as a ubiquitous client. The client does not bring data. 3D GIS data is served from the

* *Corresponding author, Email : cychoi@kangwon.ac.kr*
*Manuscript received Jun. 13, 2012; revised Feb 23, 2013; accepted Mar 03, 2013*

server instead. Requesting data is based on the current viewpoint. We use available information about the scene (camera poses and GPS data) to build a request and send to a remote server. Optimized 3D GIS models are generated from the server and sent back to the client to display.

The rest of the paper is organized as follows. Section 2 discusses the backgrounds and 3D object transformation. Section 3 illustrates the system architecture, flow chart and then describes the implementation process. Section 4 shows the demo results. The paper ends with the conclusion and future work in section 5.

## 2. BACKGROUND AND 3D OBJECT TRANSFORMATION

A mobile augmented reality application using 3D requires an optimized 3D format and a high quality 3D renderer. To demonstrate, we choose iOS platform. PowerVR's POD [7] format, a popular geometry file format for the iPhone was used due to its fast loading and optimized memory. The POD format is binary and one of the most widely used formats in OpenGL for Embedded System (OpenGL ES). POD files offer a much more optimized solution. As is Cocos3D [8] framework, free well-designed framework for building games and applications that plays out in 2D. Using Cocos3D, we can effortlessly create a full 3D game or application without getting into the basic essentials needs of the OpenGL ES state machine, and without having to switch over to C or C++, as required by most other 3D frameworks.

One of the necessary features for a realistic view of 3D model in augmented reality is 3D rotation. An average user expects that the 3D object would not remain at a fixed position on the camera. The 3D object must be rotated in the opposite direction that the user is walking and fixed on the ground. Hence, when moving around the object, the active camera must be transformed according to the device's sensing data. We extract the phone-based vectors then consider the perspective camera model to generate the transformation.

### 2.1 Phone-based Vectors Extraction

We suggest a phone-based 3D object transformation method that exploits the phone's internal sensor information. In this work, the transformation of 3D object is automatic with the use of translation and rotation. By extracting motion vectors from the phone's sensors, we can build a matrix for 3D object transformation. The mobile phones' sensor information is exploited to compose the 3D motion vectors, as shown in Fig.1.

The translation vector is extracted by measuring the acceleration in G's (gravitational force) along the three spatial axes at a moment of time (accelerometer sensor) and a position including latitude, longitude and altitude components from location sensor. A G is a unit of gravitation force equal to that exerted by the earth's gravitational field (9.81 m $s^{-2}$). The translation vector consists of three components x, y, z according to the translation along the X-axis, Y-axis, Z-axis, respectively, as shown in Fig.1a.

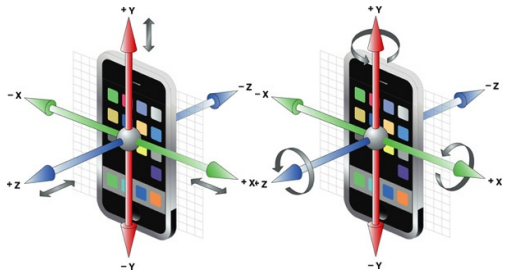



Fig. 1. Phone-based vectors extraction:
(a) Translation extraction, (b) Rotation extraction

The rotation matrix is extracted by measuring the device's rotation rate in radians per second around three axes (gyroscopes sensor) and a value H representing the heading relative to the magnetic North Pole, which is different from the geographic North Pole (compass sensor). The value H equals to 0 means the device is pointed toward magnetic north, 90 means it is pointed east, 180 means it is pointed south, and so on. The rotation matrix consists of the rotations around the X-axis, Y-axis, Z-axis, as shown in Fig.1b.

The combination of the translation vector and the rotation matrix generates the matrix for 3D object transformation.

### 2.2 Perspective Camera Model

To display 3D objects into a 2D camera view, a geometrical adjustment is required. We assume a perspective projection model, as shown in Fig. 2 to define the geometrical relationship of the 3D objects and the camera view.

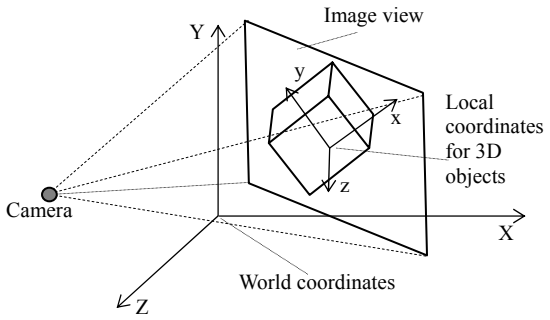



Fig. 2. Perspective projection of 3D object

Fig. 2 shows two sets of coordinates: one for world and one for local. The three rectangular unit vectors X, Y and Z of the world coordinates define 3D axes of the overall scene. The fundamental camera parameters of the position and the view direction are illustrated in the figure. The local coordinates are for the 3D objects. The 3D objects are projected in perspective onto the image view and rendered to the camera view.

The model is 3D GIS model therefore it carries information about size, location. The only other factor we need is the height of the object in real life. The ratio of the size of the object on the sensor and the size of the object in real life is the same as the ratio between the focal length and distance to the object. To work out the size of the object on the sensor, work out its

height in pixels, divide by the image height in pixels and multiply by the physical height of the sensor.

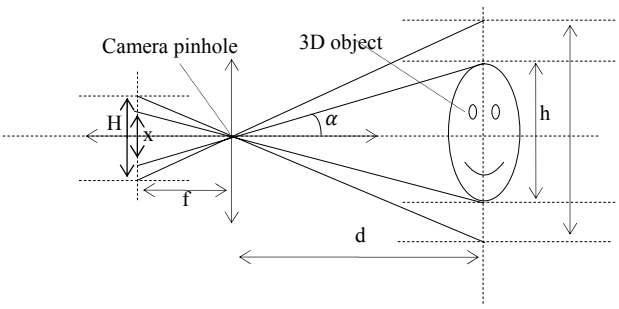



Fig. 3. 3D object onto the screen view projection

The whole sum is expressed in the Eq. 1. As shown in Fig.3 and in the equation, d is the distance to objection from the nodal point, f is focal length of the lens, h is the real height of the object, H is the image height (height of screen), x is the height of object on screen and s is the sensor height. d, f, h and s, or H and x are measured in the same units, e.g. mm and pixels respectively.

$$d(mm)=f(mm)*h(mm)*\frac{H(pixels)}{x(pixels)*s(mm)} \quad (1)$$

Increasing the focal length and keep everything else constant makes the distance increasing as focal length is on the numerator. This is what we would expect, if we have to zoom the lens to make one object the size another equally sized object used to be, the first object must be further away.

The distance again increases if we increase the real height of the object as if two objects of different real heights appear the same height in the image the taller one must be further away.

If we increase the image height, then the distance increases, as if two objects appear the same pixel size in a cropped and uncropped image then the object in the uncropped image must be further away. Two objects are the same size and remember we are keeping everything else constant.

The distance decreases if we increase the object height in pixels, as if two equally sized objects, one takes up more pixels, it must be closer.

In this work, the size of object (x) on the sensor is unknown, and the only known parameters are h (the real height of the object), d (the distance to object), H (screen height) and f (focal length of the lens). The distance to object d can be determined because the 3D object is placed at an absolute location (latitude, longitude) and we can also determine the nodal point location due to GPS sensor information. The sensor height s can be estimated only one time for every device and placed as a hardcoded constant number according to device's type. For example, to develop the application on iPhone, we place s as $s_{iPhone}$ and the constant is used in real-time without recalculating. To calculate the constant $s_{iPhone}$, we can simply keep a distance 3 meters from a blank wall which contains two sketched points. By pointing the camera to the median point of the line connecting two sketched points, we can measure the line length appearing on camera view and then easily calculate the constant $s_{iPhone}$ as shown in Eq.2 where h is the distance between two sketched points on the wall.

$$s_{iPhone}(mm)=f(mm)*h(mm)*\frac{H(pixels)}{x(pixels)*d(mm)} \quad (2)$$

Finally, we calculate unknown $x_{iPhone}$ height of object on the iPhone's sensor via the Eq. 3.

$$x_{iPhone}(pixel)=f(mm)*h(mm)*\frac{H(pixels)}{s_{iPhone}(mm)*d(mm)} \quad (3)$$

We note that we can also calculate the angle θ between the horizon line of camera view and the line connecting nodal point to object location. Then we adjust the object's image on the screen far away from the center point of screen based on the angle θ.

## 3. DESIGN AND IMPLEMENTATION

### 3.1 System Architecture

Due to memory constraints of mobile devices, a client-server architecture was chosen. The client is ubiquitous and doesn't bring 3D models data. Data is generated from server. With a request-response model using URL query string and JSON protocol over HTTP, the client sends viewpoint requests to a remote 3D model optimizer server for the model generation. Optimized 3D models are sent back to the client for rendering.

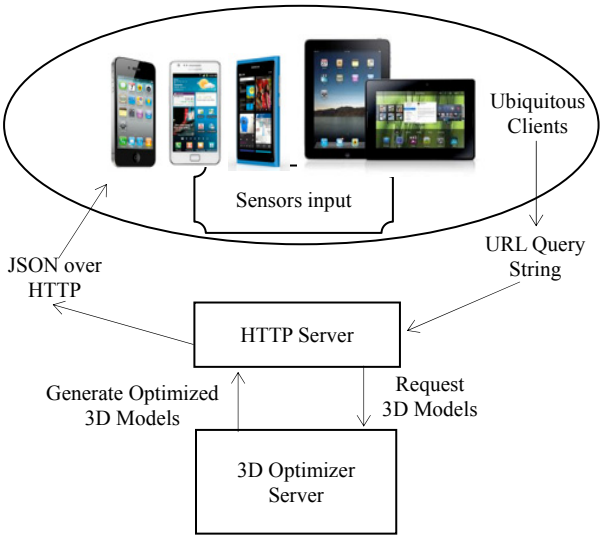



Fig. 4. Overall system architecture

As shown in Fig.4, the *ubiquitous client* displays the visualization of the requested view enabling the user making new queries based on the current location. Inputs using built-in GPS location listeners and accelerometer sensors of iOS devices allow the user to manipulate the viewpoint using geo-positioning and orientation. The client is connected to the GPS receiver from which it continuously reads GPS positioning data. According to GPS position and accelerometer change notifications, requesting 3D models for current viewpoint is generated and sent to the server. Ubiquitous clients can be different types such as iPhone, iPad, Samsung Galaxy S2, Samsung Galaxy Tab, Nokia, etc.

*The HTTP server* listening requests from clients is implemented as a lightweight bridge in Python mapping the requests from URL query string into actual C function calls.

The server provides concurrent access to the 3D optimizing server for multiple clients.

*3D optimizing server* is an implementation of the algorithm of optimizing 3D triangulations using discrete curvature analysis [9]. The 3D optimizer analyzes the requests and generates optimized models.

### 3.2 Client-side Structure

The iPhone SDK [10] has a good I/O abstraction layer instrumental in implementing the GPS connectivity library and the core motion with accelerometer support.

OpenGL ES appeared to be the only option for using real 3D on an iOS device. However, Cocos3d based on OpenGL ES is very powerful and royalty-free and there are a large number of resources available for developers. Besides, the documents of cocos3d are detail for every function. Thus, we used cocos3d framework as root development platform.

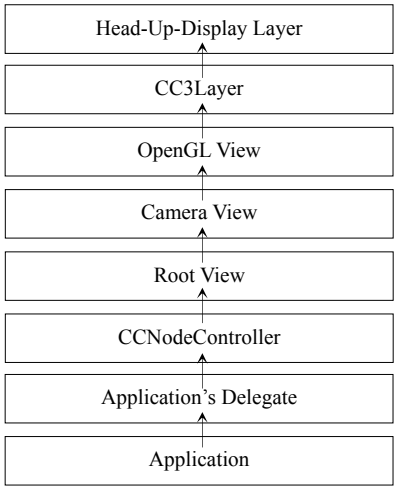



Fig. 5.Structure of client application

It is also necessary to consider how the different parts of the application are ordered. The application has an application's delegate to call a root CCNode controller developed in cocos3d. The CCNode controller includes method to add a camera view below an OpenGL ES2.0 view and a top layer (CC3Layer) with its head-up-display where other elements are displayed. It is important to set the views in correct order to make sure that one view doesn't hide another. Fig. 5 shows the structure of client application.

Client-side implementation for mobile devices is divided in six stages, as shown in Fig. 6. Stage 0 focuses on selecting the technologies and 3D model format to be used.

Stage 1 starts the implementation process with the need of initializing the cocos3D framework. We used cocos3D framework as root development platform instead of UIView family. Because UIKit cannot be managed by CC3Node (the primitive element in cocos3D class structure), it can only be managed manually. The map view control is a built-in control in iOS SDK as known with MapKit framework. This stage includes initializing MapKit layer that contains the information of map view. This layer would be called by Head-Up-Display layer (HUD layer). HUD layer is above the main layer and its scene contains a camera, light, the interface elements of radar, compass, address and location information, heading information, buttons, or any other controls that we want to show on the top of the screen view. The HUD Layer is necessary to contain extra controls and display the contents of augmented reality.

Stage 2 uses the camera through a capture session manager and the combination of the camera with rendered 3D models on screen view. The camera is setup to be used in both normal mode and recording mode. Recording mode is necessary whenever the user wants to record the current screen view. In this stage, the main scene of the system turns to schedule update mode. The inner nodes will be considered to render automatically for every frame. Initializing the location and sensor listeners is included to make sure the stage has information enough to translate to the next stage.

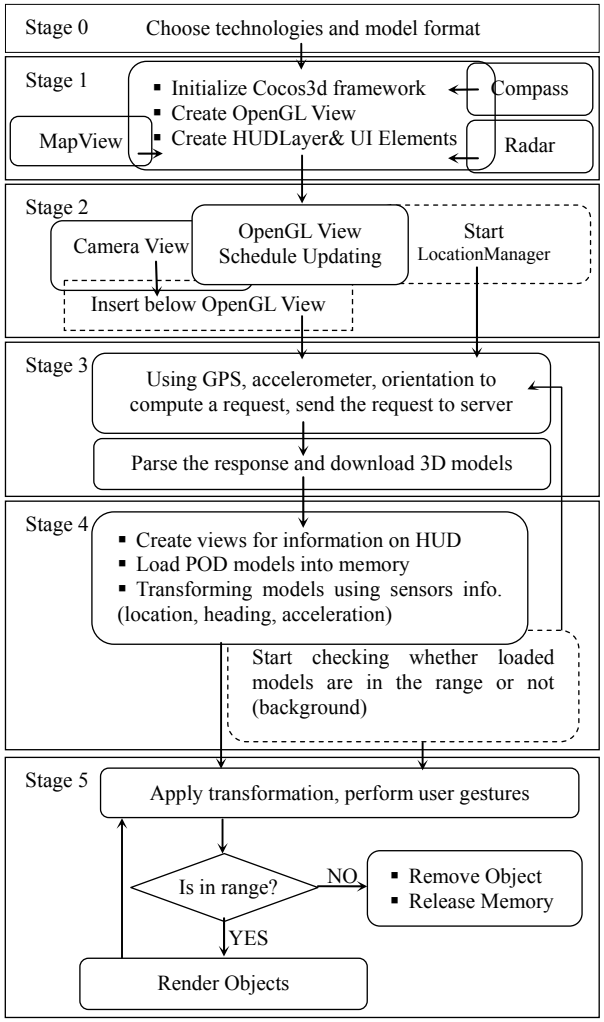



Fig. 6. Implementation processes of client

Stage 3 focuses on the use of sensors to determine the positioning of the device and its orientation. Directional information leads the system building a request to the remote server. The request includes the information where the device is and the radius of range calculated from the device's location. The extra information of the request should include short

description of the target place we want to find. The response from server turns the system to filter the contents under constraints in settings. Every result alerts the system to download necessary geo-tagged contents. They are POD files or icons of the points. This stage includes the tasks to check the valid memory and the performance at the current state. Tasks are in a concurrent executor queue with settable properties. If the number of current tasks is greater than a pre-set maximum number, the task will be in the queue and waiting for its turn. The maximum size of downloaded contents is settable in settings. In this stage, before continuing to the next stage, testing is required to confirm the valid contents if they can be loaded or not.

Stage 4 works with the loading of downloaded contents. In this stage, all necessary information including 3D models will be loaded into memory and prepared to process. The system uses the current sensors information to create markers and models inside the main scene. All of models will be transformed to the correct locations and rotations based on their locations. This stage starts the checking task to test the 3D models if the models are in the valid range or not. This testing is required before continuing to the next stage.

Finally, Stage 5 introduces the interface and the operations that need to be performed according to the user's gestures. These operations are 3D rotations, translations and zooming. The contents are now rendered into the screen view. This stage includes the display of compass changing, radar scanning, map moving and the movement of the device based on the changing of location and sensors. Apply these 3D transformations to the activated camera will tell the contents if the contents are in the valid range or not. If one content is out of range with a settable threshold number, the system will remove that content to save the performance and buffer memory. The removed contents are still stored at client until the next new contents with greater rank come. When the memory reaches the limit, the system uses the information of range, distance, location or rank to determine what content should be held on or deleted from the device.

### 3.3 Implementation

This implementation assumes that the 3D models may fix and is optimized in the direction. As a result the location of 3D model is fixed according to the real world. The movement of the device tells the system to change the transformation of the activated camera.

**3.3.1 Request – Response Model:** A stateless Uniform Resource Locator(URL) query string [11] and JavaScript Object Notation (JSON) [12] over Hypertext Transfer Protocol(HTTP)is used for the client-server interaction. The client sends a request with a URL format and the response in JSON format. The server is possibly serving multiple clients and doesn't keep the state of previous requests. The client doesn't uniquely identify itself, therefore the server cannot make any assumptions if a client has previously made any requests. A request from client is in the following URL format:

*http://samplehost.com?latitude={0}&longitude={1}&radius={2}&limit={3}&searchStr={4}*

where the parameters *{0}* and *{1}* are the current latitude and longitude, respectively, of client; radius parameter *{2}* is range of the search in meters; limit parameter *{3}* provides the limitation of the results in order to ensure the stable bandwidth; the last parameter *{4}* is the search string that can include title, zip code, address, or contact information to let the filter works and serves better results. The last parameter can be empty.

The server after receiving a request from client turns to process and sends response back to the client. Fig. 7 illustrates an example of response in JSON. *"requestId"* is the identification of current request. *"takenDate"* provides the time when taking the request and the *"results"* is list of all found contents.

```
{
    requestId:"1234567890ASDF",
    takenDate:"2012/04/11 11:12:13 1234",
    results:[{
    distance:"200",
    place:{
        location:{
            address:{
                city:"Chuncheon",
                countryCode:"KR",
                country:"Republic of Korea",
                house:"1",
                street:"Chuncheon Street""
                postalCode:"200"
            },
            elevation: "2000"
            position:{
                longitude:"37.3826103",
                latitude:"127.5044212"
            }
        },
        type:"university",
        contacts:[{
            type:"phone",
            value:"+000"
    }, {
            type:"website",
            value:http://www.kangwon.ac.kr
        }],
        placeId:"123456789009876543234567543",
        name:"Kangwon National University",
        iconUrl:"http://domain.com/res/uni/kw.png",
        podUrl:"http://domain.com/pod/kangwon.pod",
        rank: {
            average:"5.0"
        },
        sizeOfPOD: {
            value:"12345",
            unit:"byte"
        }
    }
}, {...
```

Fig. 7. An example of response in JSON format

For every updating of the viewpoint actuated by a position and orientation from an iOS device's GPS module and accelerometer sensor, a new request is generated. The request is parsed by the server and passed to the 3D optimizing server. Invalid requests are discarded if they do not adhere to the URL string format. The 3D optimizing server in turn generates the

optimized 3D models according to the positions of models in the real world. From the current location, a far 3D model based on its location will be generated with reduced mesh but a near 3D model requires a detail mesh.

**3.3.2 Server:** To handle clients with possibly HTTP requests concurrently, the server was implemented in Python using base HTTPServer package from the standard Python distribution. This base package provides the basic functionality of a multithreading listening that supports concurrent for incoming requests. The request handler using the urlparser to parse and validate the URL query string dispatches a request. The server then delegates the optimizing generation to the 3D optimizer using the Python Boost C++ interoperability library. The optimized 3D models are generated and stored in a temporary folder with a flag variable to handle the download states from clients. When the client finishes downloading generated models, these temporary models are moved into a cleaner queue and removed into the trash. Fig. 8 shows the processing sequence diagram.

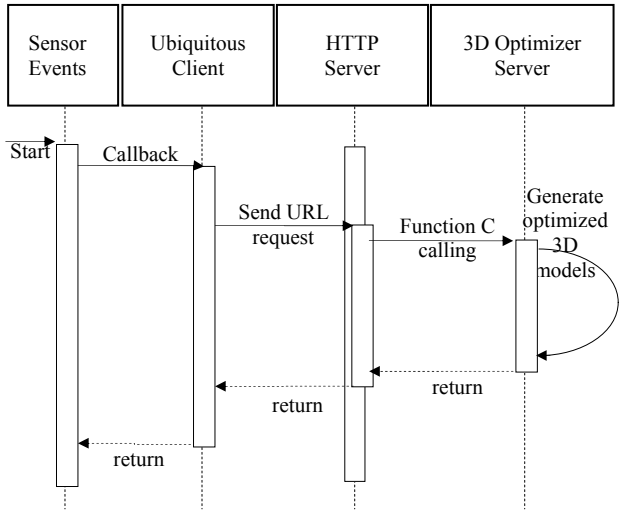



Fig. 8. Overall system sequence diagram

**3.3.3 User Interaction:** The interface includes a collection of controls such as labels to display current address and positioning information, address button to open the extra information of current location, hide/show compass button, motion mode button to enable or disable the sensors, hide/show radar button, hide/show map view button, layer button to hide or show the HUD layer and setting button. The video recording button turns the system to record what things are displaying on the current screen. The compass control shows the changing of compass information. All found places around the current location would be shown in the radar control. The map control enables an extra looking to realize the current movement.

## 4. RESULTS AND DISCUSSION

### 4.1 Results

The captures obtained from the application when moving the device around a certain location, the 3D model is rendered at center of screen. Fig. 9 shows different views of a model at the same location. The virtual model always stands beside the real object television whatever the orientations of camera are. The display is independent from the device's movement.

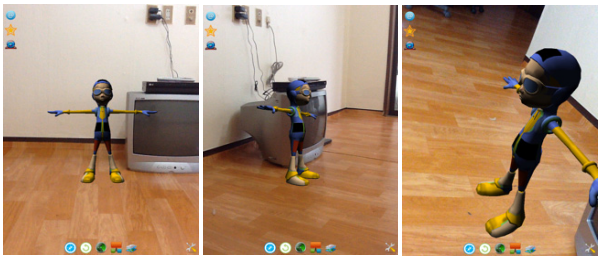
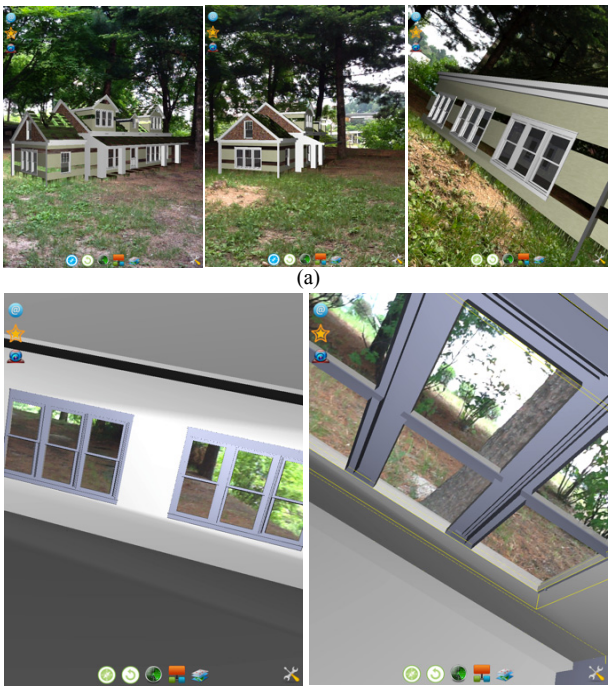


Fig.9. Different views of a model at the same location whatever the orientations of the camera are.



(a)

(b)

Fig.10. Standing around and inside a model
(a) Different views of the same house model
(b) The views inside the house model, looking outside through the windows

In Fig. 10, the model is a house. We can move around the house model or enter inside as well. As shown in Fig. 10a, we can move surrounding and view different aspects of the model. Holding the device and point the camera to a certain place where we expect to see the model. Rotating the device or changing standing place, we will see that the house model does not fix at a position on the screen. Like a real house, the model looks fixed on the ground and does not depend on our movement. The house model in the figure also has inside space. Fig.10b shows that we can enter the inside space and look around the walls or look outside through windows.

Fig. 11 shows full display of the application. The interactive controls include an address button to show the current location

in detail by using Daum Local API. The favorite button as presented in star symbol is used to bookmark the current place under a rating number. The camera button turns the screen into recording mode to record the current display. The compass button will control the compass view. Next is the button that will enable or disable device motion listeners. Whenever there is a changing of heading or location, if the motion mode is enabled, the camera's transformation will be applied and vice versa. Radar scanning and updating is based on the controlling of radar button. Next to the radar button is the button controlling the display of map view. Layer button helps us to choose what kind of contents should be displayed and what shouldn't. The setting button turns screen into configuration screen. The setting includes the radius of radar scanning, maximum threads, minimum accuracy numbers, threshold, etc.

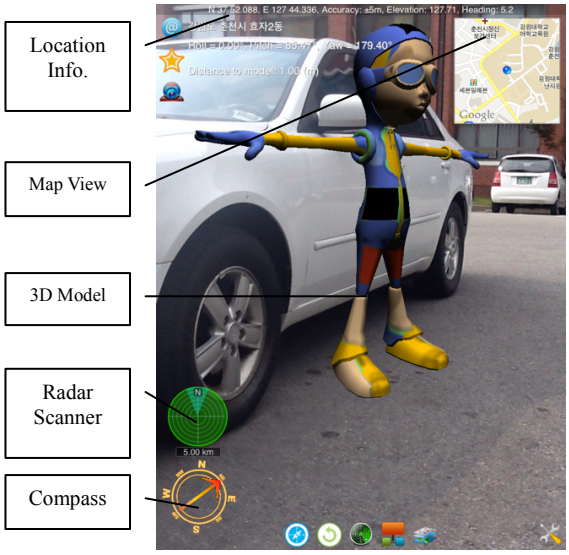



Fig. 11. Display with full controls on iPad

### 4.2 Performance

The perceived performance is directly coupled to the time between user input and the resulting change in the viewpoint including 3D rendering. This depends on the quality of the network link to the server and client's computing.

While WLAN is generally not available in the outdoors because of its limited scope, UMTS is widely deployed. The limited bandwidth should be less of a problem. Currently, insufficient or unreliable Internet access outdoor make use of this implementation difficult in practice. When WLAN is available, performance of the system is sufficient.

This system depends on the GPS system which is undoubtedly the most famous sites provide broad coverage. It has certain limitations due to radio signals sent by GPS satellites does not penetrate most buildings or even go through the thick vegetation. The impact of urban canyons is called in the city due to the reflection of GPS signals and switches to failure to locate, as aid in the control is most needed. However, GPS is a system of dominant position and alternatives such as observation cell phone tower estimated position of the user did not provide any complete comparison results and has not taken into account. In the prototype developed, there is no fallback in case of GPS outages when the satellite does not look or other failures to get a GPS signal. In case of such failure, the latest3D model is displayed if it's valid. This also applies to situations where customers fail to the server.

The horizontal accuracy, however, has also an impact on the performance of the application. Especially when the distance to the 3D model is very short, generated model data from server needs to be in detail. It requires client device spending much time in rendering task.

### 4.3 Discussion

Proposed application is giving the ability to overlay 3D models onto the physical world based on its location and camera input of iOS devices. The interaction is based on the real movement. A head-up-mounted device integrates with this work would help the augmented reality more realistic.

But it is currently not possible to manually offset the coordinates acquired from the GPS receiver. The location of the viewpoint is always relative to the actual GPS position and the direction is based on the device's sensor. In the ideal conditions where the GPS signal works smoothly with high accuracy, the application runs effectively. In contrast, the application shows errors if the signal from the GPS is inaccuracy. Moreover, the view containing the 3D model is limited. To get the total view of a large 3D model, users need to move out from target point far enough to view all aspects of the models.

## 5. CONCLUSIONS

We explore the feasibility of ubiquitous devices to possibly enable visual display and crisper at unknown location and enhance the concepts of orientation in mobile augmented reality application. We focus on detecting GPS location and orientation in the augmented reality spatial 3D representation of the user's geographical surroundings.

This paper proposed a new method for augmented reality display. With the geo-located information and accelerometer sensing, a 3D model can be transformed to correct location and displayed on the view of camera without using any markers. When moving camera around a certain 3D model, the transformation is updated based on the changing of heading, GPS location and device motions accelerometer. The 3D objects used in the system are 3D GIS models which carry positioning information themselves. The system with only necessary loaded and rendered contents based on the current location is an instance of a location-based application. We believe this system provides a compelling challenge for tourism development strategies.

In the future, we are planning to improve the positioning accuracy and to conduct a fast rendering to deal with big size 3D models. We also plan to implement the client system on other mobile platforms such as Android, Window Mobile.

## REFERENCES

[1] Chang, W.; Qing Tan, "Augmented Reality System

Design and Scenario Study for Location-Based Adaptive Mobile Learning," 2010 IEEE 13th International Conference onComputational Science and Engineering (CSE), 2010, pp. 20 – 27.
[2] Hincapie, M.; Caponio, A.; Rios, H.; Mendivil, E.G., "An introduction to Augmented Reality with applications in aeronautical maintenance," 2011 13th International Conference onTransparent Optical Networks (ICTON), June 2011, pp. 1 – 4.
[3] Nate Hagbi, Oriel Bergig, Jihad El-Sana, Mark Billinghurst, "Shape recognition and pose estimation for mobile augmented reality,"Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality, 2009, pp. 65 – 71.
[4] J.T. Doswell, M.B. Black, and J. Butcher-Green, "Mobile Augmented Reality System Architecture for Ubiquitous e-Learning," Fourth IEEE International Workshop on Wireless, Mobile and Ubiquitous Technology in Education, 2006. WMUTE '06, Nov. 2006, pp. 121 – 123.
[5] Jungsik Park Byung-Kuk Seo and Jong-Il Park, "3D visual tracking for mobile augmented reality applications," IEEE International Conference on Multimedia and Expo (ICME), July 2011, pp. 1 – 4.
[6] Liestøl, G., "Situated Simulations: A Prototyped Augmented Reality Genre for Learning on the iPhone," International Journal of Interactive Mobile Technologies iJIM, 3, 2009.
[7] Imagination Technologies Ltd. PowerVR Insider Utilities. http://imgtec.com/powervr/insider/powervr-utilities.asp.
[8] The Brenwill Workshop Ltd. The Brenwill Workshop. http://brenwill.com/cocos3d
[9] Kai Hormann, Sun-Jeong Kim, and David Levin Nira Dyn, "Optimizing 3D Triangulations Using Discrete Curvature Analysis," Mathematical Methods for Curves and Surfaces, 2000, pp. 135–146.
[10] Developer. iOS Dev Center. https://developer.apple.com/devcenter/ios/index.action
[11] Wikipedia. Query String. http://en.wikipedia.org/wiki/Query_string.
[12] Introducing JSON. http://www.json.org
[13] Azuma, R. T., "A Survey of Augmented Reality". Media, 6(4), vol. 6, 1997, pp. 355-385.
[14] Wither, J., DiVerdi, S., and Höllerer, T., Annotation in outdoor augmented reality, Computers & Graphics, Elsevier, 33(6), 2009, pp. 679-689.

**Phuc Vo**
He received the B.S. in computer sciences and engineering from University of Technology, Vietnam in 2010 and also received M.S.in computer and communications engineering from Kangwon National University, Korea in 2012. His main research interests include information system, information security, data storage, data mining, geography information system, computer graphics, and visual communication.

**Chang Yeol Choi**
He received the B.S, M.S in electronics engineering from Kyungpook National University, in 1979, 1981 respectively and also received Ph.D. in computer engineering from Seoul National University in 1995. Before joining the School of Computer Science and Engineering, Kangwon National University in 1996, he has been with Electronics and Telecommunications Research Institute (ETRI) during 1984-1996, where he served as a principal researcher of Computer Research Division. His research interest lies in the field of computer architecture, embedded system, and ubiquitous services.