2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing

# *Reconstruction of 3D Maps for 2D Satellite Images*

Lianzhang Zhu
College of Computer & Communication Engineering
China University of Petroleum
Qingdao, China
zhulz@upc.edu.cn

Xuexing Zheng
College of Computer & Communication Engineering
China University of Petroleum
Qingdao, China
cocos2dphone@gmail.com

Pengfei Li
College of Computer & Communication Engineering
China University of Petroleum
Qingdao, China
lpfupc@gmail.com

***Abstract*—In this paper, we will propose a new approach of building 3D maps from 2D satellite map images. Firstly, we get a 2D satellite map image from Google Earth. Then we use OpenCV to extract the elements' boundaries automatically on the 2D map image. Next, we correct the automatic extraction result and set the information of the elements on the map manually. After extraction and setting is finished, we will get the data (elements' boundaries, types, locations, textures, heights) of the elements on the 2D map. With these data, we can establish the virtual 3D digital map of the scenes. At the same time, we will add textures to the 3D models in the 3D scenes. To be more user-friendly, the system provides data gloves based roaming and can synchronize the 3D scenes with Android clients.**

***Keywords- satellite image; 3D map; 2D map; roaming***

## I. INTRODUCTION

At present, three-dimensional digital map can be divided into real 3D digital map and virtual 3D digital map. Based on shooting or laser scanning to the real objects, the real 3D digital map is achieved mainly through the photographic image processing and measurement. This kind of map is superior to the two-dimensional map for being intuitive, accurate and information plenty enough. However, because of the large amount of information, high investment, and strongly limited application range, it is not easy to be widely applied. The virtual 3D digital map is based on the real geographic information. There are many methods of building this kind of map, such as CAD technology, OpenGL technology and GIS-based software technology [1]. The virtual 3D digital map has a small amount of data, friendly interface, fast reaction speed, and it is mainly applied to the city's 3D visualization.

This paper presents a new approach to construct 3D map from 2D map: firstly, we get a 2D satellite map image from Google Earth; then, we extract the buildings' boundaries automatically on the two-dimensional satellite map image with OpenCV [2] technology; next, we amend the extraction results, and set the elements' information to obtain data related to the establishment of virtual 3D digital map; at last, the 3D map can be generated with these data which are obtained before. To be more user-friendly, system provides data gloves based roaming and can synchronize the 3D scenes with Android clients.

The synchronization between 3D scenes and Android clients can help identify a person's location in the 3D scenes. If one is in trouble in the scenes, it could be helpful for the person. The research is useful. On one hand, it provides real scene reference for tourists. On the other hand, this paper provides a good approach to urban planning; it not only saves resources, but also brings considerable economic benefits.

In Section II, we will show an establish procedure overview of this paper's approach. In Section III, we will show the approach implementation in detail with an example of the establishment of part of China University of Petroleum campus. In Section IV, we will show the results of system's performance evaluation in the initialization of 3D scenes; it shows the relationship between response time of initializing 3D scenes and the number of 3D models in the 3D scenes. We conclude in Section V with some related work. Finally, we make some conclusions about this paper's approach and our future work.

## II. APPROACH OVERVIEW

### *A. Rational behind 3D map reconstruction from 2D maps*

When we browse the satellite map, we can find more detailed information about the map, such as the contours of buildings, roads, lawns and forests, if we zoom the satellite map to the maximum level. In most cases, the contours are quite obvious. Therefore, we can use the OpenCV technology to identify these contours, and we will get the boundaries of the elements. Therefore, the boundaries of buildings, roads and other elements in the satellite map can be identified. After finishing the identification of the boundaries, we can set the information (such as the locations, types, textures and heights) of the buildings, the roads and other elements. Then we can establish the corresponding scenes of virtual 3D map according to the information obtained. According to this principle, an approach of establishing virtual three-dimensional map from two-dimensional map is proposed in this paper.

### *B. Approach Target and Process*

Through the combination of OpenCV technology and OpenGL technology, the two-dimensional satellite map images are converted into the corresponding three-dimensional scenes of virtual three-dimensional map, which is a relatively realistic

978-0-7695-5046-6/13 $26.00 © 2013 IEEE
DOI 10.1109/GreenCom-iThings-CPSCom.2013.298



IEEE computer society

Authorized licensed use limited to: b-on: UNIVERSIDADE DE AVEIRO. Downloaded on October 01,2025 at 22:47:53 UTC from IEEE Xplore. Restrictions apply.

3D virtual scene for roaming. The approach's implementation (Figure 1) is described as follows:

Firstly, we photograph landscapes and landmarks with a digital camera in the real scene, and transfer relevant data from the camera to the computer which converts these pictures into PNG format. These pictures will be used as textures for the 3D models in the generated 3D scenes. Besides, we get a 2D satellite map image from the Google Earth, and store it as PNG format, and take it as the input of the map analysis program.

Secondly, by processing the 2D satellite map image with OpenCV technology, we can get data of the elements which need to be built in the 3D scenes. These data include elements' boundaries, types, heights, locations and textures. These data will be sent to the 3D map generation program through the Internet.

Next, the 3D map generation program receives and analyzes these data which are taken as the input data for 3D scenes' generation. Then the 3D digital map is created with the usage of these data. The 3D scenes' terminal not only supports the keyboard roaming, but also supports data gloves based roaming. In addition, 3D terminal and Android clients can communicate with each other through the UDP protocol. So it is convenient to identify the location of the one who uses the application on an Android cellphone in the 3D scenes.

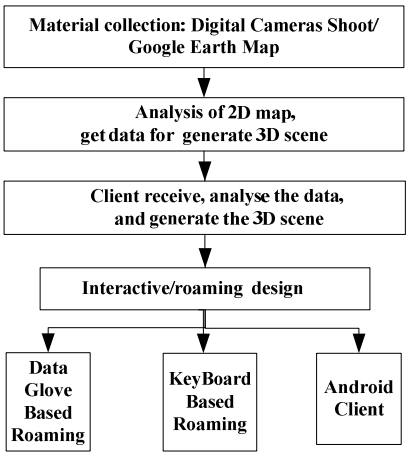


Figure 1. Process Procedure Diagram of System Design

## III. IMPLEMENTATION

The implementation of this approach mainly includes four sections: data collection, establishment of 3D roaming scenes, roaming control and communication between 3D terminal and Android clients. A brief introduction about the four sections is given below.

### *A. Data Collection*

Data for constructing 3D scenes come from the analysis of image processing. The operation procedure is as follows:

STEP 1 Import 2D PNG map image into map analysis program.

STEP 2 Identify elements' boundaries of 2D map with OpenCV technology, and get preliminary contours of the elements' boundaries in the 2D map. The main process procedure is shown as Figure 2.

STEP 3 Correct the automatic extraction result and set the corresponding information of buildings (such as: type, height, texture material and location) manually.

STEP 4 Save the data and send it to the 3D client through the Internet manually, and the data is taken as the data source for generating 3D map.

In most cases, the elements' contours on the map are relatively easy to identify, but sometimes contours may be connected together which leads to imprecise recognition, that's why we need manual correction on STEP 3.

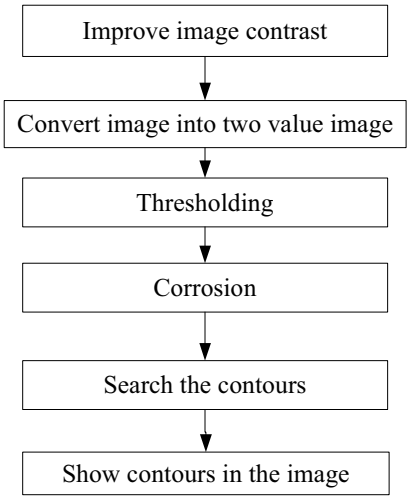


Figure 2. Process Procedure of Contours Extraction

The result of automatic contours extraction is shown at Figure 3. From it, we can see that the extracted elements' contours are not very accurate. This is because: in order to facilitate the construction of elements in the 3D map generation terminal, we set the extracted elements' contours as rectangles.

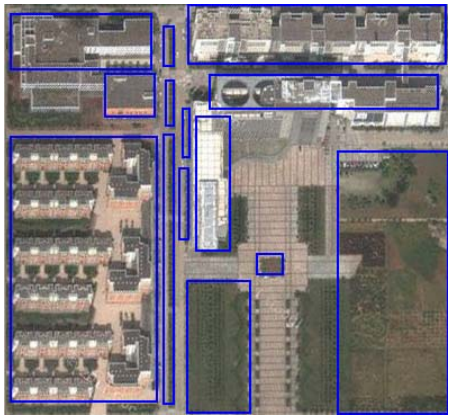

Figure 3. Result of Automatic Contours Extraction



Authorized licensed use limited to: b-on: UNIVERSIDADE DE AVEIRO. Downloaded on October 01,2025 at 22:47:53 UTC from IEEE Xplore. Restrictions apply.

Figure 4 is the interface of regional adjustment. In it, the *"MaxRect"* and *"MinRect"* set the contour size limits of extracted elements, "*MaxRecst"* is the upper limit, and *"MinRect"* is the lower limit. *"Extract"* execute the function of extracting the map's elements automatically. *"AddElement"* is adding a new element to the extracted elements. When we double-click an element area, dialog box of element's region information will appear, and the interface is shown in Figure 5. In this interface, we can set the information of elements, such as types, heights, textures and locations. When the setting is finished and *"OK"* is clicked, the information of an element can be stored into the corresponding temporary variables.

After all the settings are finished, if we execute the *"Save to TXT"* function in Figure 4, the data will be preserved into a *"txt"* file by the program according to specified storage format. The data is taken as the data source for the generation of 3D scenes.

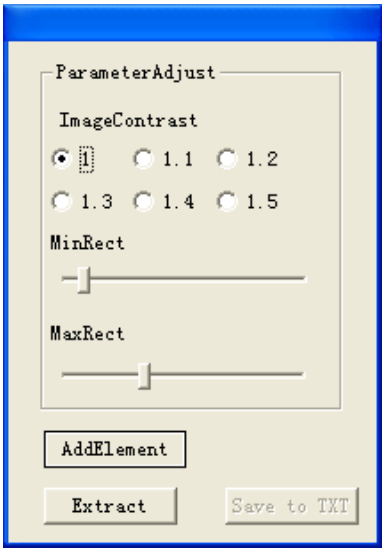


Figure 4. Regional Adjustment Interface

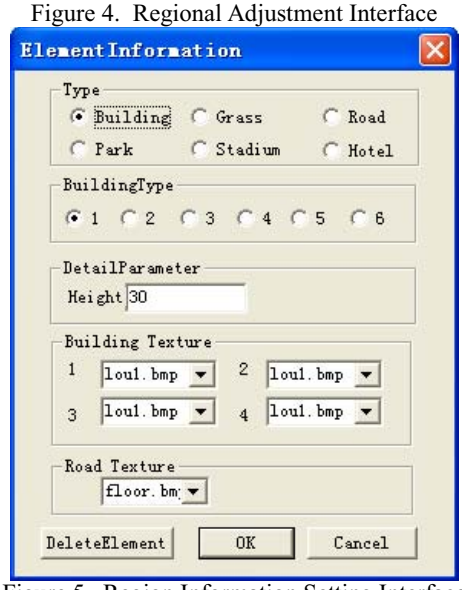


Figure 5. Region Information Setting Interface

## B. *Construction Of 3D Scenes*

Through the usage of OpenGL rendering technology and the data in the *"txt"* file, we can build the 3D scenes. Then, a relatively realistic virtual three-dimensional map is provided for the user. The operation procedure is as follows:

STEP 1 The scenes' data file is opened and the elements' information in the file is read automatically by the program.

STEP 2 The data in the file is analyzed and the elements' information (number, type, texture, height, location) in the scene is got automatically by the program.

STEP 3 The elements of the resolved is drawn one by one automatically by the program. When all the rendering is completed, we can get the corresponding virtual three-dimensional map scenes.

After the 3D scene is generated, the users can browse the details of a place with first person perspective view (Figure 6) or overlook the scenes (Figure 7).


Figure 6. Effect of First Person View


Figure 7. Effect of Overlook

In Figure 6, the upper right corner shows the effect of 2D map in the corresponding 3D scenes. Only in the first person view, one can roam in the 3D scenes. If a person is roaming in the 3D scenes, his location in the corresponding 2D map will also change which ensure the person's location in 2D map and 3D map is consistent. In addition, the upper left corner of Figure 6 shows the current state of character (forward, backward, turn left, turn right), and the top of Figure 7 shows the current system time.

In the 3D scenes, the display of current view region is achieved through "*void gluLookAt (GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble centerx, GLdouble centery,*



Authorized licensed use limited to: b-on: UNIVERSIDADE DE AVEIRO. Downloaded on October 01,2025 at 22:47:53 UTC from IEEE Xplore. Restrictions apply.

*GLdouble centerz, GLdouble upx, GLdouble upy, GLdouble upz);"*. And the *(eyex, eyey, eyez)* specifies the position of the eye point, the *(centerx, centery, centerz)* specifies the position of the reference point, the *(upx, upy, upz)* specifies the direction of the up vector. The function uses the principle of perspective to show the current scene. When we are roaming in the 3D scene, the key code of coordinate transformation is as follows:

```
Forward:    eyez += (float)sin(rad) * speed;
            eyex += (float)cos(rad) * speed;

Backward:   eyez -= (float)sin(rad) * speed;
            eyex -= (float)cos(rad) * speed;

Turn Left:      s_angle -= 2.0;

Turn Right：    s_angle += 2.0;
```

Where: *"rad"* represents radian, *"speed"* represents the forward amplitude which is a fixed value; *"s_angle"* represents angle. In order to cancel the distorted phenomenon of visual effect, when we are roaming in the 3D scene, we put the 3D scene in the sphere environment. Therefore, when we are roaming, in order to adapt to the coordinates of the characters in a 3D scene, a conversion of angle is needed.

Figure 8 is system class diagram of elements' types in the 3D scene. It can be seen from the figure that the elements in the system include the following several types: *ball, cube, cone, rectang, cylinder, trees*. Among them, the 3DS tree model is used to load tree models and the *rectang* is rectangle which is used for laying on the ground or the lawn. All of these types are inherited from the 3DObeject *class*; there are several basic methods in the parent *class*:

- *Read()*: Read the data of the elements, such as number, types, textures, heights and locations;
- *Save()*: Set the positions of elements in the current three-dimensional coordinate system.
- *SetParam()*: Before rendering the 3D scenes, calling this function to initialize the elements' positions in the current three-dimensional coordinate system;
- *LoadTexture()*: load corresponding textures when loading the 3D models;
- *Draw()*: Draw models in the three-dimensional scenes.

In addition, the program also provides some 3DS models, for example: *trash, car, hotel, stadium* etc.

The storage structures of 3D models are as follows:

```
class CVector3 //Save the vertex of a 3D Model
{ public:
    float x, y, z;
};
class CVector2   //Save the UV Texture Coordinate of Model
{ public:
    float x, y;
};
struct tFace                    // The structure of Model Face
{       int vertIndex[3];   // Index of the Vertex
        int coordIndex[3];  // Index of the Texture
Coordinate
};
struct tMatInfo  // The structure of Texture
{       char  strName[255]; // Name of Texture
        char  strFile[255];  // Name of Texture File
        BYTE  color[3];   // The RGB Color of Object
        int   texureId;   // ID of Texture
        float uTile;      // Repeat of U
        float vTile;      // Repeat of V
        float uOffset;   // Offset of Texture U
        float vOffset;   // Offset of Texture V
} ;
struct t3DObject    //Information structure of 3DObject
{       int numOfVerts;   // Vertex Numbers of a Model
        int numOfFaces;   // Faces Numbers of a Model
        int numTexVertex; // Texture Numbers of a Model
        int materialID;      // ID of Texture
        bool bHasTexture;  // Exist Texture Mapping or Not
        char strName[255];  // Name of Object
        CVector3  *pVerts;  // Pointer of Model Vertexes
        CVector3  *pNormals; // Normal Vector of Object
        CVector2  *pTexVerts; // Coordinate of UV Texture
        tFace *pFaces;   // Information About Model Faces
};
struct t3DModel             //Structure of 3D Model
{       int numOfObjects;      // Object Numbers of Model
        int numOfMaterials;   // Material Numbers of Model
        vector<tMatInfo>pMaterials; // Material List
        vector<t3DObject> pObject; // Object List
};
struct tChunk //Structure of Chunk Which Save Information
{       unsigned short int ID;     // ID of Chunk
        unsigned int length;       // Length of Chunk
        unsigned int bytesRead; // Read bytes of Chunk
};
```

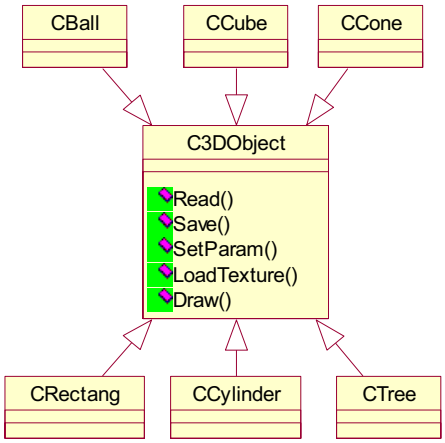


Figure 8. Types of 3D Models



Authorized licensed use limited to: b-on: UNIVERSIDADE DE AVEIRO. Downloaded on October 01,2025 at 22:47:53 UTC from IEEE Xplore. Restrictions apply.

### C. *Phone Client Terminal*

The phone client can communicate with 3D map by UDP protocol, so that we can locate the man who is using the application with an Android cellphone at the 3D terminal's scenes. The Android client terminal also can keep pace with the 3D terminal. The interface of the Android simulator is shown as Figure 9. The red balloon on Figure 9 is the current location of the man in the 2D map.

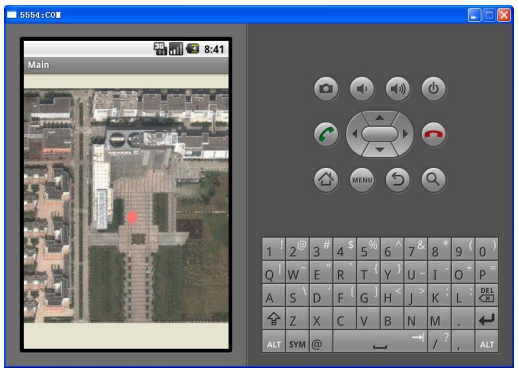


Figure 9. Interface of Android Simulator

### D. *Data glove Based Roaming*

At first, we achieved the roaming function by camera with OpenCV technology on the 3D digital map. To begin with, the program gets the video through camera. Then, it locates the hand by analyzing the video. Next, it recognizes the gesture and responses to the gesture. Because of the similarity of the color of human skin, the camera is single and the algorithm is not good enough, there is a big problem in locating user's hand exactly which resulted in a bad effect on user experience.

To improve the users' experience in roaming, we implied the function with data gloves [3], instead of using OpenCV technology. There are several pressure transducers in the data gloves. When we bend our fingers, the transducers can get the pressure value and send the values to the program. And the program will recognize the defined gestures according to the values. After the gesture is recognized, the program will call the corresponding functions of 3D scenes' change.

## IV. System Evaluation

The environment of system evaluating is *GHOST XP SP3* operation system; the processor of the computer is *Pentium(R) Dual-Core CPU T4200*; the dominant frequency of *CPU* is *2.00 GHz*; the available memory is *1.87 GB*. And the evaluating result is shown at Table 1.

Table 1. Result of System's Performance Evaluation

| Models' Number in the 3D Map | Response Time(S) | |
|---|---|---|
| | Initialize Scenes | Refresh Data |
| 100 | 0.875 | 0.844 |
| 200 | 1.410 | 1.344 |
| 300 | 2.250 | 2.214 |
| 400 | 2.720 | 2.655 |
| 500 | 3.512 | 3.442 |

The time provided in Table 1 is the average of ten times' measurement. And the corresponding line chart is shown as Figure 10. Because the OpenGL is real-time rendering, we do not evaluate the response time of roaming.

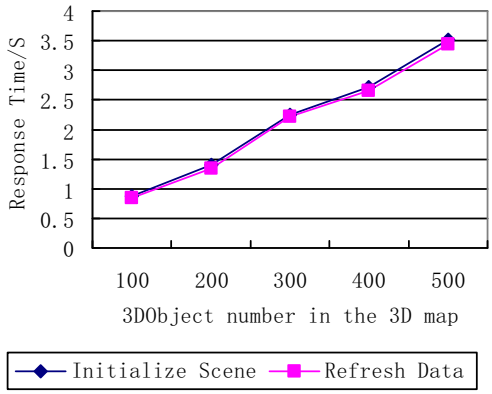


Figure 10. Line Chart of System's Evaluation Result

From Table 1 and Figure 10, we can find that with the increase of models' number in the 3D map, the response time is increasing approximate linearly, and the response time of initializing 3D scenes and refreshing data are almost the same.

During the process of system's evaluation, we find that when the models' number in the 3D map achieves 600, the system's performance takes a sudden turn and becomes worse rapidly. But when the models' number is 500, everything is ok. The main reason for this is as follows:

- The program adopts OpenGL technology, and rendering the models one by one. When we rendering too many models at the same time, the rendering speed will be affected.
- The performance of the computer itself may also have an effect on scenes' rendering, for example: the CPU processing speed, the running of other program.

## V. Related Work

Yang Yanhong et.al [4] established the 3D digital map of Guilin University of Electronic Technology campus. To begin with, they established a scenario framework with 3DS MAX and Auto CAD technology. After adding texture to the framework with images, light rendering, entity baking, they get the finished 3D digital map. The approach in [4] has something in common with this paper's approach. They all need collect scene materials, including collecting scenes' images with digital camera and 2D map from Google Earth. They all need to process the materials with Photoshop. But they are different in the way of building 3D models of the scenes. The 3D map generated in [4] is more realistic in detail, because its model framework is built with 3D MAX which can be more detailed, while this paper's approach generates models with OpenGL rendering, it is not easy to draw an model with too many details. For example, the steps of a building can be created easily with the 3D MAX software when we creating a scenario framework, but it has difficulty in drawing the steps with OpenGL when we are rendering a number of buildings. This paper's approach



Authorized licensed use limited to: b-on: UNIVERSIDADE DE AVEIRO. Downloaded on October 01,2025 at 22:47:53 UTC from IEEE Xplore. Restrictions apply.

generates simple regular 3D models, and for the complex irregular Object, we make approximation replacement with small 3DS models. The approach in paper [4] is not universal, and it is complicated to modify the finished 3D digital map when the scenes are changed. This paper's proposal is more universal, easy to operate and simple to modify when the scenes are changed. Besides, the approach in paper [4] limits the path of automatic roaming and it is lack of flexibility.

Based on the existing 2D vector data, the approach in [5] introduces the city 2D building's attribute *"height"* as elevation data, and constructs the simulated 3D map service. The paper [5] mentions that their method is rapid, flexible, simplify and enhances people's sense of geospatial cognition. But it did not show any detailed information about the system's performance evaluation. It also mentions that the surface of building lacks of realistic. From the given result, we can also find that only 3D buildings are generated in paper [5] which means that the lawn and forests are ignored when they are building the 3D map.

With the usage of photogrammetric technique, the approach in [6] generates part of three-dimensional model buildings of Universiti Teknologi Malaysia campus and classifies the buildings based on categories using photogrammetric data and technique. Take it as an example, the paper [6] compares several special software (Erdas Imagine, SpacEyes3D Builder, Google SketchUp) in building 3D models. First, they obtain sets of measurements (length, width, and height) of selected buildings. Then, they process the input data (two overlapping aerial photographs covering part of Universiti Teknologi Malaysia campus camera parameters, and ground control points) separately with the above three softwares, and compare the result they get. After comparison, they find that Erdas Imagine is better than the other two softwares in accuracy. Though these softwares have good accuracy in generating 3D models from 2D map, they can't make texture rendering according to the real scenes.

Similar with the work of paper [4], the approach in [7] first generates 3D model framework with 3D MAX. Then, it adds textures with bmp images by using OpenGL technology. From the given result, we can find that the effect is not bad. It mentions that the method is effectively without showing any detail information of system's performance evaluation.

With the technology provided by SketchUp and ArcGIS 9.3 and several other softwares, the work in [8] generates 3D digital map of Yangtze University campus. Before making 3D digital map, they build a 2D electronic map of the scene, where they take the object that needs to build models with SketchUp as a point layer. They provide the special function of 3D attribute query, and make a secondary development on the 3D map that is built with SketchUp. The given result is good, but it doesn't add image textures to the buildings, and the paper doesn't show any detail information about roaming and system's performance evaluation.

The relate works above only focus on one aspect of the generation of virtual 3D digital map with the existing software, and none of them combines the virtual 3D digital map with phone client. Apart from the difference in building 3D scenes, the synchronization between 3D client scenes and Android clients is also a special work of this paper.

## VI. CONCLUSIONS AND FUTURE WORK

This paper shows a different and feasible approach of generating 3D digital map from 2D map. Compared with the work of [4-8], this paper's approach not only supplies 3D digital map generation, but also supplies data gloves based roaming and synchronization between 3D client scenes and Android clients. Therefore, we can locate a man in the 3D map scenes. Compared with real 3D map, its effect is not good but could be more widely used with lower costs.

In the future, we will future investigate related graphics algorithm, to improve the extraction result and recognize more complex contours of elements on the 2D map. Besides, we will improve the rendering algorithm to increase the number of rendering elements that the program can at the same time, and enhance the ability of rendering more complex models with OpenGL. What's more, more 3DS models will be provided to increase the alternative ability of models for different scenes. We will modify the contour extraction algorithm, so that we can get the height information about the buildings according to the scale between the buildings' length and height on the 2D map.

## ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to professor Weishan Zhang and professor Yong Li.

## REFERENCES

[1] Congshu, S., & Rui, X. (2010, March). Development Status and Implementation Research of Three-Dimension Digital Map. In Education Technology and Computer Science (ETCS), 2010 Second International Workshop on (Vol. 2, pp. 641-643). IEEE.

[2] Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media, Incorporated.

[3] 5DT Data glove Ultra Series. October 2004. (In Chinese)

[4] Hongyan, Y., Zhuo, S., & Yanru, Z. (2011). Design of digital campus roaming system based on virtual reality technology. Journal of Guilin University of Electronic Technology, 4, 008. (In Chinese)

[5] Xiao, W., Yehua, S., Yongning, W., Jingwei, S., & Hongping, Z. (2010, December). Research on simulated three-dimensional map service from two-dimensional city building data. In Information Science and Engineering (ICISE), 2010 2nd International Conference on (pp. 3950-3953). IEEE.

[6] Ahmad, A., & Rabiu, L. (2011, March). Generation of three dimensional model of building using imagegrammetric technique. In Signal Processing and its Applications (CSPA), 2011 IEEE 7th International Colloquium on (pp. 225-231). IEEE.

[7] Zhang, L., Chen, M., Liu, Y., & Li, Y. (2011, September). The Method to Construct the Three-Dimensional Model Based on OpenGL. In Information Technology, Computer Engineering and Management Sciences (ICM), 2011 International Conference on (Vol. 3, pp. 140-143). IEEE.

[8] Wang, Q., & Zhu, X. (2010, November). The implementation of campus 3D electronic map based on SketchUp and ArcGIS. In Audio Language and Image Processing (ICALIP), 2010 International Conference on (pp. 1031-1034).IEEE.



Authorized licensed use limited to: b-on: UNIVERSIDADE DE AVEIRO. Downloaded on October 01,2025 at 22:47:53 UTC from IEEE Xplore. Restrictions apply.