ATLANTIS PRESS

Advances in Engineering Research, volume 140

2nd International Conference on Electrical, Control and Automation Engineering (ECAE 2017)

Data Analysis System of Oblique Photography Based on OpenSceneGraph

Lei Shi[1], Xiaopan Zhang[1,*], Xiaoyan Ma[1], Kai Ye[1] and Yaowu Liu[1]

[1]School of Resource and Environmental Engineering, Wuhan University of Technology, Wuhan, 430070, China

[*]Corresponding author

***Abstract*—Oblique photography technology is a new earth observation technology which is developed in the international field of surveying and mapping in recent years, and it can reflect the features of the real situation of the surrounding. Besides, oblique photograph can greatly promote the application of remote sensing image. In this paper, the oblique photograph data is used as the object of research, and the OpenSceneGraph (OSG) interface is used to process it. We first completed the LOD loading and displaying of oblique photograph data, then, we realized the data view scene roaming and operation using OSG library. Finally, we completed the display of the attribute information of the object in the oblique photography data using the OSG library and C++ program language. The system has achieved the functions finally, and through the test and verification, the result of the system is basically reliable. It has laid a good foundation for the further research of the oblique photograph data.**

Keywords-oblique photograph model; OpenSceneGraph; display of the attribute information

I. INTRODUCTION

In recent years, with the rise and development of the concept of smart city, more and more workers are willing to invest in 3D scene technology. Oblique photography technology is playing an increasingly important role in urban 3D. The oblique photography is a new technology of earth observation, which has been developed in the field of international surveying and mapping in recent years [1]. When the tilting image is automatically modeled, the resulting model is an Irregular Triangulation Network (TIN) with texture map, and it is not divided into single selected objects based on buildings. Without the division of the oblique model, the GIS management and application cannot be further applied.

The technique of oblique photography is developed by the photogrammetry. Oblique photography is by carrying multiple sensors on the same flight platform, at the same time by an ortho four inclined multi angle image acquisition [2], at the same time to obtain a positive investment image can get around the image and building height, the side texture attribute information, which is greatly facilitate the data's get. The oblique photography technology makes the image more suitable for human eye observation, but also obtains the accurate location information of the object through advanced location technology. Wider and richer image information has also promoted the development of remote sensing technology.

Oblique photography technology in China began to develop in recent years. The group of academician Liu Xianlin succeeded the first domestic oblique camera SWDC-5 on research and development in October 2010, and in the same year his group and Beijing Eastdawn company carried out cooperation Changchun City oblique photography project successfully, which achieves the official sales of domestic oblique cameras [3]. After nearly five or six years of development, many companies have invested in the development of oblique photography technology, and have made many breakthroughs. Aero photogrammetry is also moving towards to without the ground control points [4]. In August 2013, the Skyline V6.5 products launched by Beijing Terry skyline Technology Co., Ltd. increased the oblique photogrammetry, batch automatic panoramic 3D modeling technology and mobile internet terminal products, thus triggering the domestic oblique photography 3D modeling technology climax. Beijing SuperMap Software Co. Ltd. produces SuperMap (2015) GIS 9D, break through the difficulties oblique photography using the model, and provides a set of powerful oblique photography 3D model application solutions, which opened a new chapter in the field of 3D GIS data sources. Wuhan international navigation company gRob R2 series, GeoPro V2 series, GeoPro K2 series products, Beijing HOPONG company Micro UAV oblique photography system, Guangzhou Hi●TARGET company iScan series and so on can improve the efficiency and accuracy of data acquisition of data in a certain extent [5].

The development of oblique photography data analysis system in this paper is mainly aimed at the result of automatic modeling of oblique images acquired by UAVs, that is, files in .osgb format. The realization of the property information acquisition function of the format file can improve the dynamic and flexibility of the image of the oblique photography to a certain extent.

The goal of this paper is to implement a simplified oblique photography data analysis system based on OSG library, including LOD data layered loading display, scene roaming and operation, object attribute information display. The realization of these functions can play a very good role in the development of the later oblique photography data analysis system and the application of the oblique photography technology.

II. METHOD

A. Technical Route

The data of obliquei photography technology is usually collected by carrying oblique cameras on UAVs, and then

processed by automatic modeling software, and the real 3D model of .osgb format is generated [6]. In our current environment, the demand for the application of the computer 3D digital model is rapidly increasing [7].

OpenSceneGraph is a high performance open-source 3D graphics engine, which is based on the interface of the software industry standard OpenGL and C++ developed in recent years and is an open source and cross platform graphics development kits, it can also be said to be a basis of advanced simulation software [8], and make developers more convenient to study and use interactive graphics program.

The main function of the oblique photography data analysis system is to use the OpenSceneGraph library to roam and operate the data based on the shortcomings of the previous oblique photography data processing. Before analyzing and processing data, the use of OpenSceneGraph library at PC needs to configure and compile the running environment of the library, and then we can develop and process the system based on Visual Studio 2012 software platform. In this system, we first realize the LOD data hierarchical load and display including a scene and a plurality of scene data. On this basis, we can achieve the scene roaming and related operations and obtain .osgb file texture. The related operations include operation response event OSG library itself and obtain three-dimensional coordinates of the mouse click. We can calculate the texture coordinate with the same index by 3D coordinates which is the point value of the texture, which can be combined with texture so we can obtaine the image pixel value in the texture. Finally, we can get the pixel color value, which means we can achieve the acquisition of attribute information.

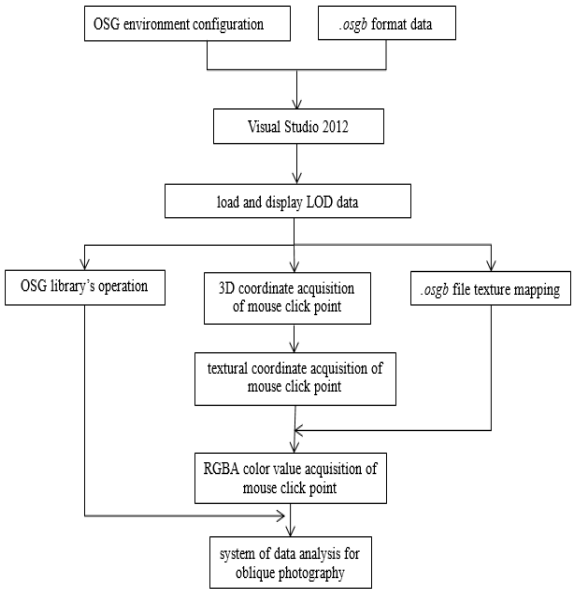

FIGURE I. THE TECHNICAL ROUTE OF THE DATA ANALYSIS SYSTEM OF OBLIQUE PHOTOGRAPHY MODEL

B. Layered Load and Display of Osgb Data

Oblique photography data's the most basic operation in the hierarchical display referring to the essential premise of the header file defines a scene for viewer, then we use the ReadNodeFile method of osgDB to read the .osg or .osgb model file to the node, and then use the setSceneData method to set up the scene to the root node of all scene data. Finally, we implementation of viewer and circular rendering can be loaded into the model file.

C. Acquisition of the Color value of Mouse Click point in the Scene

The principle of mouse click point color value acquisition is to get the pixels corresponding to mouse click points in texture mapping, which requires first obtaining the texture map of the model. The acquisition of pixel values can be expressed by formula (1).

$$\begin{aligned} s &= S * tex\_x; \\ t &= T * tex\_y \end{aligned} \tag{1}$$

Among them, s and t are pixel s axis and t axis coordinate respectively. S represents the length of s axis of texture map, T represents the length of t axis of texture map, and tex_x and tex_y represent the texture coordinates of the point respectively. The acquisition of texture coordinates requires mapping transformation relation between texture maps of the points on the surface and the triangle. The transformation relation is calculated from the 3D coordinates and texture coordinates of the three vertexes of the triangle surface, so we can get the textural coordinates by the 3D coordinates of the points and the transformation relation.

To achieve the above functions, first of all, we want to make a simple description of the coordinate and texture coordinates defined by the OSG library. The OpenGL points use row vector, and the points in OSG library use column vector, but coordinate system defined in OpenGL for the X axis on the screen to the right and the Y axis on the screen in the vertical axis and Z vertical screen outward, so OSG library only along the OpenGL coordinate system's X axis direction of the screen rotated 90 degrees, that is coordinate system in OSG to X axis is on the screen to the right and the Y axis is perpendicular to the screen and the screen of Z axis is in the vertical coordinate, which meets right hand coordinate system standard[9]. The geometry of 3D scene is usually made up of vertices, but to achieve the real display of an object, texture is needed on the surface of the geometry. The texture supported by OSG library includes six kinds of texture expression methods: one-dimensional texture, two-dimensional texture, two-dimensional texture array, three-dimensional texture, cubic texture and rectangular texture [12]. This paper uses two-dimensional texture mapping method. The two-dimensional texture type is rectangular. When the two-dimensional texture is projected onto a non-rectangular geometry, it needs to be converted, and the two-dimensional texture coordinates need to be set manually. The two-dimensional texture coordinate system belongs to the Descartes coordinate system. Usually, three axes of S, T and R are used to express the X, Y and Z axes of the world coordinate system, and the range of the texture coordinates is (0.0,1.0) [10].

1) Texture mapping: In the texture mapping, a two-dimensional texture Texture2D class is used to map the texture and add it to the texture properties of the geometry by using the StateSet class, such as:

```
osg::StateSet* stateset = new osg::StateSet;
osg::Texture2D* texture = new osg::Texture2D;
texture->setImage(image);
stateset->setTextureAttributeAndModes(0,texture,osg::StateAttribute::ON);
geometry->setStateSet(stateset);
```

So when texture mapping is obtained, we first need to get the rendering state of the model StateSet, then get the texture attribute of the state, and get the texture map of the model by getImage.

2) Acquisition of 3D coordinates of mouse click point in the scene: After the 3D coordinates of the mouse click point in the scene are obtained, the 3D coordinates of the point can be output at the console. The realization of the function needs to create a new class in the osgGA *GUIEventHandler* class, and then use the PUSH asscee in *osgGA* class *GUIActionAdapter* class to catch mouse events, and through the intersection function of *LineSegmentIntersector* class in *osgUtil* library to get the point of intersection, and use the *getWorldIntersectPoint()* method to obtain three-dimensional coordinates of the intersection point, such as:

```
osgUtil::LineSegmentIntersector::Intersections::iterator hitr = intersections.begin();
point_ = hitr->getWorldIntersectPoint();
```

The output of the coordinate values can be displayed by using the cout method of the namespace *std*.

3) Acquisition of mouse click point textural coordinates in the scene: The texture coordinates are converted from the three-dimensional coordinates of the model points. The use of *getVertexArray()* and *getTexCoordArray()* methods of Geometry class in OSG Library can obtain 3D coordinates and texture coordinates of all vertices, and then we used *getNumElements()* method to obtain the number of 3D vertex and textual vertex, those are equal, which could prove that the 3D vertex array and textural coordinate array is one-to-one match. The texture coordinates value of the vertex can be obtained by obtaining the 3D coordinates of the same index as the vertex. The non vertex texture coordinates are obtained by data transformation.

The reconstruction of the oblique photography data model needs to redraw polygonal faces, most of which are triangles. The acquisition of the non vertex texture coordinates can be obtained by matrix operation between three textural vertex corresponding to the three vertex coordinates and the three vertex coordinates of the triangle faces. All triangle vertex lists of the object can be obtained by using the OSG library's triangulation function TriangleFunctor template. According to the sum of the angles between the three vertices of the triangle and the connection between the intersection points, all triangles can be traversed to determine which triangle of the intersection is in [11]. Then the texture coordinates corresponding to the three points are obtained according to the index values of the three vertices of the triangulation. The texture coordinates are two-dimensional, so the Z value can be set to 0. Set the first triangle vertex v1 (x1, y1, z1) , which corresponds to the texture coordinates for t1 (tx1, ty1, 0), the second v2 (x2, y2, z2), which corresponds to the texture coordinates for t2 (tx2, ty2, 0), the third v3 (x3, y3, z3), which corresponds to the texture coordinates for t3 (tx3, ty3, 0), texture mapping matrix of the triangle is A, with the following formula (2) relationship:

$$\begin{pmatrix} \mathrm{tx}\,1 \\ ty1 \\ 0 \end{pmatrix} = A \begin{pmatrix} x1 \\ y1 \\ z1 \end{pmatrix} \tag{2}$$

Similarly, for v2 and t2, v3 and t3, all have such a relational expression. However, due to the matrix of 3*1 cannot be inverse, the computation of transformation relation of matrix A is relatively complicated. Therefore, three points can be used for computation at the same time, as shown in formula (3),

$$\begin{pmatrix} tx1 & tx2 & tx3 \\ ty1 & ty2 & ty3 \\ 0 & 0 & 0 \end{pmatrix} = A \begin{pmatrix} x1 & x2 & x3 \\ y1 & y2 & y3 \\ z1 & z2 & z3 \end{pmatrix} \tag{3}$$

among them,

$$A = \begin{pmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{pmatrix}.$$

In this way, the affine transformation relation of texture A can be obtained. The texture coordinates of the point can be obtained by clicking the three-dimensional coordinates of the point of the mouse.

Learning and researching the classes and methods used to get the three vertex coordinates from the mouse click event to the triangle face of the mouse click point, finally, we find the index information om the slice element of the mouse click point that can be directly get. The specific method is: when we get the intersection of ray emitted by mouse click and model, the IndexList type vertex index form in Intersection structure in LineSegmentIntersector class can access by using the IndelList method of the osgUtil library, which can get the three vertex index values by the mouse click in the triangles, thus we can get the texture coordinates of the mouse click point.

4) Acquisition of color values: After obtaining the texture map of the model file and the texture coordinates of the mouse click point, we can get the pixel value of the mouse click point in texture mapping by formula (1), and then use the *getColor()*

method in the *Image* class to get the RGBA color value of the mouse click point.

int teximage_x = (int)((teximage->s())*(tx));

int teximage_y = (int)((teximage->t())(ty));*

osg::Vec4 teximage_color =

teximage->getColor(teximage_x,teximage_y,0);.

III. EXPERIMENT RESULTS

This system deals with the analysis of the .osgb format of the oblique photographing data file *Tile_+000_+000_L22_0000001.osgb*. The principle is that by clicking any point in the scene with the mouse, the triangle corresponding to the point can be found in the texture map by using the texture coordinates of the three vertices of the triangle where the point is located. By verifying whether the corresponding point of the mouse click point is in the texture map Triangle tablets to determine the accuracy. Due to the experiment was processed by using the mouse to click on the judgment, so in a test can only verify a point, the results are shown in Figure 3.1. Click on the location of the arrow in the scene to get the corresponding position of the mouse click point in the corresponding texture image and obtain the attribute information. Further validation of the experiment shows that any point in the clicking scene can get the corresponding point of mouse click point and get the attribute information of the point in the texture map of the scene.

FIGURE II. THE RESULTS OF THE EXPERIMENT

The operation results show that within a certain error range, the processing and analyzing result of the system is acceptable, what's mean is that the system can be applied to the research of the attribute information acquisition of mouse click point in oblique photography data.

IV. CONCLUSION

Through the use of .osgb format oblique photography data model file on the oblique photography data analysis system for testing and verification, we can get available results, that is within a certain error allowable range, the system through the mouse click to get the attribute information is basically correct, and the RGBA value of the color is accurate. So it can be said that the system is basically feasible in obtaining the attribute information in the mouse interaction event of oblique photographic data, which lays a solid foundation for the subsequent realization of oblique photographic data.

ACKNOWLEDGMENT

This research was financially supported by the Fundamental Research Funds for the Central Universities (175208001).

REFERENCES

[1] Ye Tian, Yu Xiang, Feng Gao, Liang Gao, Automatic and Fast 3D True City Production Based on Pictometry Oblique Photogrammetry Technology, vol. 2. Bulletin of Surveying and Mapping, 2013, pp.60-61.

[2] Li Yan, Jun Cheng, Automatic Texture Mapping of 3D Reconstruction Based on Oblique Imagery, vol. 2. Remote Sensing Information, 2015, pp.31-32.

[3] Anfu Li, Zhengxiang Zeng, Xiaoming Wu, The Analysis of the Development of Oblique Photography Technique in China, vol. 37. Geomatics & Spatial Information Technology, 2014, pp.57-58.

[4] Xiuxiao Yuan, Some Investigations of Image Orientation in Aerial Photogrammetry, vol. 22. ADVANCES IN EARTH SCIENCE, 2007, pp.828-834.

[5] Yuping Sun, Yabing Fan, Rui Hao, Bowen Qiang, Geomatics & Spatial Information Technology, vol. 38. 2015, pp.152-153.

[6] J. Salvi, C. Matabosch, D. Fofi, J. Forest, A review of recent range image registration methods with accuracy evaluation, vol. 2. Image and Vision Computing, 2007, pp.578-596.

[7] AHMED M, HAMRUNI. The use of oblique and vertical images for 3D urban modeling, Nottingham:The University of Nottingham, 2010.

[8] Yanchun Shen, Youhong Zhu, Li Cao, Zhuanping Wen., Design and Implementation of 3D Simulation Platform Based on OSG, vol. 24. COMPUTER SIMULATION, 2007, pp.207-208.

[9] Wei Rao. Global 3D scene platform based on Scenario system and multimission planning system, Huazhong University of Science and Technology. 2012.

[10] Xinxin li, Wanggen Wan, Li Li, Ximin Zhang ,Chao Gan,Xiaoqing Yu, Using OpenSceneGraph to simulate the dynamic flooding effect, Internal Conference on Audio, Language & Image Processing, 2012, pp.1155-1156.

[11] P Liu, YC Li, W Hu, XB Ding, Segmentation and Reconstruction of Buildings with Aerial Oblique Photography Point Clouds, vol. 40.The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2015, pp.109-109.

[12] RA Pavlik, JM Vance. VR JuggLua: A framework for VR applications combining Lua, OpenSceneGraph, and VR Juggler, Workshop on Software Engineering & Architectures for Realtime Interactive Systems, 2012, pp.29-30.