EFFICIENT AND ACCURATE FUSION OF MASSIVE VECTOR DATA ON 3D TERRAIN

Zhendong Liu[1]*, Chengming Li[1], Zhanjie Zhao[1], Dong Zhang[2], Fei Wang[1], Ying Wang[1]

[1] Institute of Cartography and Geographic Information System, Chinese Academy of Surveying and Mapping, Beijing, China - lzdgis08@126.com, cmli@casm.ac.cn, 807551588@qq.com, wf946773243@gmail.com, fuermoying_1993@163.com
[2] Tai'an City Golden Land Surveying and Mapping Company, Shandong, China - zhangdong890620@163.com

Commission IV, WG IV/8

KEY WORDS: vector data; 3D terrain; seamless fusion; multi thread; level of detail

ABSTRACT:

This paper presents a viewpoint-related fusion method of massive vector data and 3D terrain, in order to superpose the massive 2D vector data onto the undulating multi-resolution 3D terrain precisely and efficiently. First, the method establishes an adaptive hierarchical grid spatial index for vector data. It will determine the geographic spatial relationship between vector data and the tiles of 3D terrain in the visible area; secondly, this method will use the improved sub-pixel graphics engine AggExt to generate textures for vector data that has been bound to terrain tiles in real time; Finally, considering that a large amount of vector data will generate a lot of 2D textures in the computer memory, the method should release the "expired" vector textures. In this paper, in order to take into account the real-time convergence and the smooth interactivity of 3D scenes, this method will adopt a multi-threading strategy. The experimental results show that this method can realize the real-time and seamless fusion of massive vector objects on the 3D terrain, and has a high rendering frame rate. It can also reduce the aliasing produced by traditional texture-based methods and improve the quality of vector data fusion.

1. INTRODUCTION

In recent years, with the continuous development of computer technology and geographic data acquisition method, and the deepening of GIS applications, more and more people are demanding to deal with problems in real three-dimensional space (Li et al. 2011). In the field of 3DGIS, 2D vector data is still an indispensable data type. It mainly includes three types: points (place names, trees, etc.), lines (roads, rivers, etc.) and surfaces (lakes, vegetation, etc.). With the development of multi-resolution data acquisition technologies such as remote sensing imagery and elevation, the fusion and application of 2D vector data, 3D topographic surface models, and global remote sensing imagery have become a general trend (Kang et al. 2009). Therefore, it is one of the important problems in the field of 3DGIS how to overlay a large amount of 2D vector data on an undulating 3D terrain and ensure real-time, high-efficiency, and high-quality visualization.

Currently, the methods for superimposing 2D vector data on the surface of 3D terrain can be roughly divided into three types: geometry-based rendering method, shadow-based rendering method, and texture-based rendering method (Wang et al. 2013).

1) Geometry-based rendering method. It represents the 2D vector data as a geometric model and attaches the geometric model to the surface of the 3D terrain. So as to avoid the situation where the vector data traverses the 3D terrain and achieves that the vector geometry model can completely blend with the undulating terrain, new vertices or line segments must be introduced or deleted in the vector geometry model according to the elevation value of the terrain in the corresponding location. The cross section is shown in Figure 1. Foreign scholar Rui et al. (Rui et al. 2004) used static terrain data in a small area to establish a terrain model based on a regular quadrilateral grid, and mapped the vector data to the terrain grid using projection and interpolation, but he did not explain the amount of terrain data used in the experiment; Wartell et al. (Wartell et al. 2003) proposed a method to plot 2D line type of vector data on multi-level terrain. Because the resolution of the 3D terrain is dynamically changing according to the viewpoint, in order for the vector data to perfectly match the current terrain surface, the geometric nodes of the adjusted vector line must be recalculated.

2) Shadow-based rendering method. Inspired by the idea of shadows in graphics, Schneider et al. (Schneider et al. 2007) first realized the superposition of vector data and 3D terrain. The basic idea is to create a pixel-level precision vertical projection of vector data in 3D terrain, extend the polygon along the lowest point of the terrain to construct a polyhedron, and calculate the screen space intersection of the polyhedron and the terrain, generate a mask in the template cache of GPU and draw it on 3D terrain. The method can obtain a higher rendering quality, but all the

* Corresponding author

vector data must be extended into a polyhedron in advance, and these polyhedrons must be drawn twice. Therefore, it is difficult to achieve real-time efficiency in large-scale vector data drawing (Cao et al. 2013).

3) Texture-based rendering method. The basic idea of this method is to rasterize vector data to generate a 2D texture, and then map the texture to 3D terrain so as to achieve the seamless overlay of vector data on the terrain surface (Hong et al. 2010). The cross section is shown in Figure 2. Since this method does not need to perform geometry matching calculation processing related to terrain complexity, it overcomes the deficiencies of Geometry-based rendering method, but its main shortcomings is that the limited texture resolution can easily cause a problem of texture aliasing when the 3D terrain is enlarged. In order to solve the problem, scholars proposed a dynamic vector texture rendering method. For example, Kersting et al. (Kersting et al. 2002) used frame buffering technology to dynamically draw vector data into a 2D texture, and used a real-time texture pyramid technique to draw a high resolution texture to adopt the close viewpoint. This method can greatly save the texture overhead and time wasted scheduling. However, texture aliasing can occur at steep terrain because of the different resolutions of vector textures. Li Rong et al. (Li et al. 2011) used depth segmentation and screen space segmentation to cut up the scene volume and proposed a multi-cascade drawing method, but the efficiency was low and the rendering frame rate was not high yet.

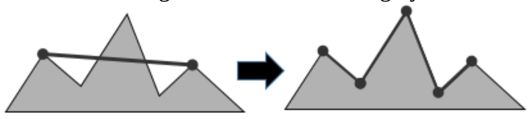


Figure 1. The Geometry-based rendering method

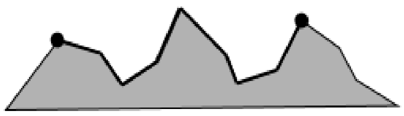
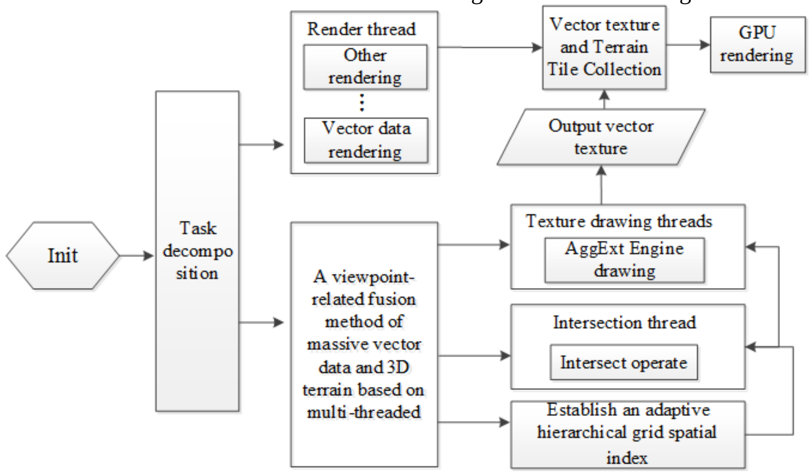


Figure 2. The Texture-based drawing method

On the whole, in the most common multi-resolution 3D terrain, the traditional several types of vector data fusion rendering methods usually have a large amount of computation and cannot achieve a better fusion effect. In particular, when faced with massive vector data, most of the methods consume too much time and cannot achieve the purpose of real-time fusion.

2. THE METHOD PROPOSED IN THIS ARTICLE

As with most graphics algorithms, texture-based rendering method will bring major problems such as texture aliasing (Wang et al. 2003) and real-time rendering. This paper presents a viewpoint-related fusion method of massive vector data and 3D terrain based on multi-thread, which mainly includes the establishment of spatial relationship between 2D vector data and 3D terrain tiles, vector objects' textures real-time rendering, and vector objects' textures released mechanism, as shown in Figure 3. Different from previous texture-based rendering methods, this paper uses the improved texture sub-pixel rendering engine AggExt to generate textures of vector data without using OpenGL's Frame Buffer Object (FBO) (Green et al. 2005) and calculating coordinate of textures. At the same time, the method uses multi-thread parallel processing technology to implement the step-by-step, phased implementation of the entire fusion process, so that when millions or even millions of vector data are superimposed on 3D terrain, it can be combined efficiently and seamlessly, and it can reduce the degree of texture aliasing of traditional methods.


Render thread
Other rendering
Vector data rendering
Vector texture and Terrain Tile Collection
GPU rendering
Output vector texture
Init
Task decomposition
A viewpoint-related fusion method of massive vector data and 3D terrain based on multi-threaded
Texture drawing threads
AggExt Engine drawing
Intersection thread
Intersect operate
Establish an adaptive hierarchical grid spatial index
Figure 3. The multi-threaded viewpoint-related fusion framework

2.1 Establish relationships between 2D vector data and 3D terrain tiles

In order to quickly find vector objects that intersect with the tiles of terrain, an adaptive hierarchical grid spatial index must be established for massive vector data (Wang et al. 2003).

After an adaptive hierarchical grid spatial index is established, combined with the 3D scene clipping, the method of this paper needs to dispatch the vector objects in the visible area into the computer memory, and then perform the operation of spatial intersection with the 3D terrain tiles. Because this article uses a quad-tree structure to organize 3D terrain. The hidden spatial relationship between the "father" tiles and their four "children" tiles can be used. According to the principle that "father" tiles do not intersect with a vector object, all its "children" tiles must not intersect with the object. It can significantly improve efficiency. Specific procedures are described below:

Step 1. To sequentially acquire the block ($Block_n$) where the vector objects are located. $Block_n$ is the current block;

Step 2. With Block$_n$ as a processing unit, the circumscribed rectangle of the Block$_n$ will be spatially intersected with the root tiles of terrain. If intersected, go to the next step; otherwise, get the next block Block$_{n+1}$ as the current block (n=n+1), perform *step 2* again;

Step 3. Traverses each vector object within Block$_n$. It will determine if the vector object intersects with root tiles. If true, the tile will bind the vector object. In particular, the vector object does not need to perform the intersection operation of the child of the root tile in advance, which can greatly save CPU computing time.

Step 4. Traverse the currently rendered tiles named Tile$_i$. Then acquire the vector object that has been bound to "father" tile of Tile$_i$, and then determine whether the vector object intersects with Tile$_i$. If true, the vector object will be bound to Tile$_i$. Taking into account that millions of vector objects participate in the intersection operation, in order to not affect the real-time and smooth interaction of the 3D scene, the task is put into a separate thread to complete.

2.2 Draw textures of viewpoint-related vector objects

At present, for the problem of texture aliasing, scholars have made improvements and achieved better results. However, there are still major problems as follows: First, most texture-based methods need to use OpenGL's FBO to dynamically generate textures and calculate textures' coordinates for vector objects in real time. A vector object will be dynamically drawn to the FBO's color buffer, then the color buffer will be given to the texture object which should be as a texture of terrain. There is no doubt that this will bring a heavy burden to the GPU, because the GPU renders the vector object twice. Secondly, these methods must calculate the texture coordinates. When a large number of vector objects need to be frequently drawn into a signal 2D texture, they will bring a lot of calculations to the CPU, and it is also difficult to ensure that the sampling density is uniform and some pixels of the texture are stretched.

2.2.1 Real-time texture drawing framework for vector objects. After the spatial relationship are established, each tile of terrain can obtain Intersect Vector Objects (IVO). The next major work is to draw the IVO into the 2D textures. Here we propose real-time texture drawing framework for vector objects.

In the 3D terrain scene, according to the viewpoint-related LOD control rules (Yang et al. 2010), IVO will follow the dynamic changes of terrain to draw textures of vector objects with different resolutions. Therefore, under the premise that there is no need to increase the pixel resolution of the texture, the pixels of each texture can be used more effectively. In addition, we propose the real-time texture drawing framework for vector objects, as shown in Figure 4 which uses an improved sub-pixel graphics engine AggExt, combined with multi thread parallel processing and view frustum cutting techniques. The main flow of the framework is as follows:

Step 1. The render thread sends drawing requests. According to the dynamic change of terrain, the IVOs in the visible area will be marked for drawing texture, and then these IVOs will be put into a request list in priority order;

Step 2. Adopt a multi-threading strategy. According to the request list and the computer CPU cores number to determine the optimal number of thread parallelism and start threads;

Step 3. AggExt engine draws textures. Each thread of texture drawing in the active state will call AggExt to perform texture drawing work. According to the vector type (point, line, surface, etc.), texture object size, the bounding box of tile and other parameters to determine the final drawing of the texture size and quality;

Step 4. The tiles bind and render textures. After AggExt generates the textures, it passes them to the render thread and binds them as an ordinary 2D textures to the associated the tiles. Then, it shares texture coordinates with the remote sensing imagery and sends them to the GPU rendering pipelines to complete the fusion rendering.

Different from other texture-based rendering methods, the method proposed in this paper no longer needs to use complex formulas to calculate the texture coordinates of the vector data's texture based on the parameters such as model view matrix and projection matrix in the terrain scene. It can significantly improve the efficiency of fusion.

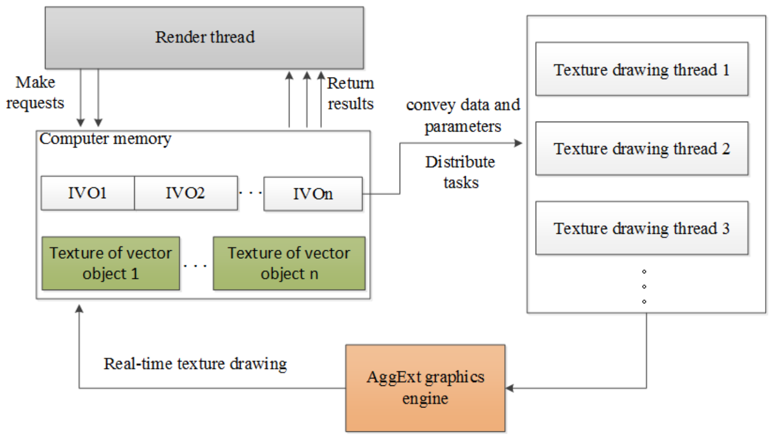


Figure 4. Real-time texture drawing framework for vector objects

2.2.2 Graphics rendering engine AggExt. It can be seen from real-time texture drawing framework that we need an independent graphics engine instead of directly using the GPU to draw textures of vector objects, thereby reducing the GPU's rendering burden effectively. When choosing a graphics rendering engine, we consider the major factors such as rendering efficiency and the minimization of texture aliasing, cross-platform. After sufficient experimental comparison, the subpixel rendering engine Agg (AGG is an open source, efficient cross-platform 2D graphics library.) was selected. The experimental conclusions are: 1) as shown in Table 1, the experiment uses Agg and the most widely used GDI+ graphics engine (Xue et al. 2015) to draw 100000 point, line, and surface objects, respectively. The latter is 2~3 times longer than the former; 2) the aspect of texture anti-aliasing. When comparing a line texture that is only a few pixels wide, it can be clearly seen from Figure 5 that the edges of the lines drawn by Agg are smoother and the effect is better.

Graphic Engine	Agg	GDI+
Point Type	1252ms	3247ms
Line Type	6255ms	14149ms
Polygon Type	3151ms	5492ms
Anti-aliasing ability	Subpixel anti-aliasing	Ordinary anti-aliasing

Table 1. Agg and GDI+ performance comparison

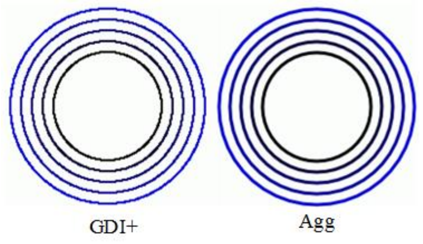


Figure 5. Anti-aliasing contrast

A lot of experiments have found that when using the Agg to draw a line with a certain width, the texture of the high-resolution will appear missing. To solve this problem, this paper analyzes and improves Agg's line object drawing algorithm.

Algorithm description and problem analysis: (1) Calculate the parameters of line texture。First of all, calculate geographic range and texture ratio of tiles: $TextureRatio = TextureSize \big/ \sqrt{(X_{\max} - X_{\min}) * (Y_{\max} - Y_{\min})}$, in the formula, TextureSize generally consistent with the size of the tile imagery texture, (Xmin, Ymin, Xmax, Ymax) is the geographic extent of the tile; Second, the line texture width: $TextureLineWidth = LineWidth * TextureRatio$, where LineWidth is the user-specified line width (for example, the specified railway line width is 2 meters). It can be concluded that the higher the resolution of the tiles (the smaller the geographic range), the greater the value of the TextureLineWidth; (2) The TextureLineWidth and vertex coordinates are transmitted to the Agg, through the stages of Vertex Source, Coordinate Conversion Pipeline, Scanline Rasterizer, and Render. The algorithm will eventually generate a 2D raster texture. During the rasterization scan phase, when sampling textures based on TextureLineWidth, the sampling interval step is determined by TextureLineWidth and Agg's predefined line_half_width. Therefore, the greater the TextureLineWidth, the larger the step, so fewer pixels can be stored in the sampling texture array (Render Buffer Array). Therefore, there is a problem of texture missing in the red line box in Figure 6.

Solution: In order to solve the texture missing problem, it is sufficient to effectively interpolate and sample the texture pixels. It proposes to use the associative container instead of the Render Buffer Array, and use the effective sampling interval value as the key value of the container, and store the sampled pixel of the texture into the container to form a Render Buffer Map; At the same time, the pixels of the texture should be interpolated and sampled according to the number of key values of the sampling container, instead of the number of sampling intervals as the termination condition. Figure 7 shows the improved algorithm drawn texture.

2.3 The textures of vector objects release mechanism

Taking into account the massive vector objects will generate a large amount of textures in real time, in order to avoid the textures growth caused by computer memory overflow. We designs and achieve the textures of vector objects released mechanism. As shown in Figure 8, only the textures of the vector objects in the VP range (the RS part) needs to be fusion-rendered; for the ES parts on both sides of the figure, if there are textures of the vector objects, they should be put into Waiting Buckets and are released at the appropriate time; In addition, in order to avoid the 3D scene is not smooth roaming, we will be distributed to a separate thread to deal with the task.

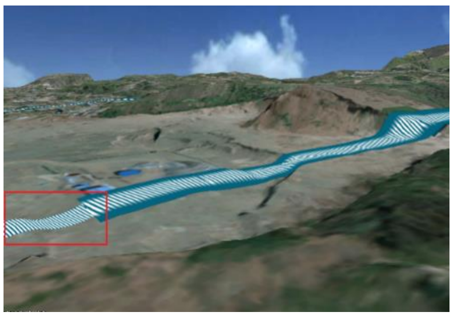


Figure 6. Fusion of line vector object at high resolutions

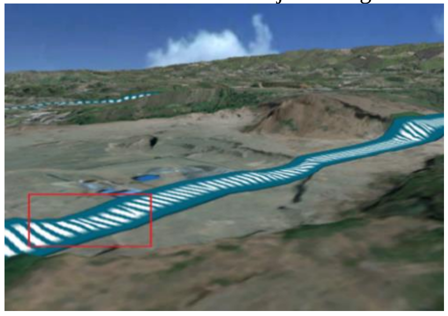


Figure 7. The correct fusion of high resolution

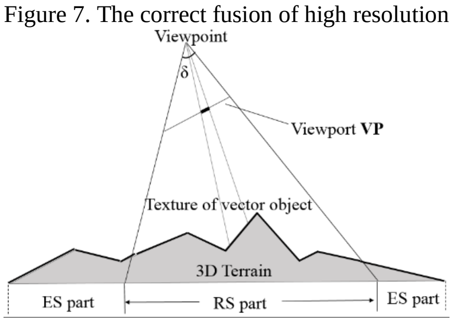


Figure 8. View frustum of vector objects

3. EXPERIMENTS AND ANALYSIS

This article relies on the NewMap software platform developed by the China Academy of Surveying and Mapping, and achieved the fusion method of this paper. It also performs experimental verification and analysis from various aspects. The experiments adopted global elevation and remote sensing imagery as basic data, and various types of vector data such as rivers, hospitals, and meadows in China were used as experimental verification data.

As shown in Figures 9 to 11, the method of this paper seamlessly merges point, line, and surface vector data with undulating terrain. Figure 9 shows the result of the fusion of point data of the national hospital with 3D terrain. Figure 10 shows the results of the fusion of meadow surface vector objects and multi-resolution terrain. It can be seen that there is no crossing phenomenon. The two graphs in Figure 11 show that as the point of view extends from the entire southwest region of China to the Chengdu region, the terrain scene gradually shows the texture of river lines with different resolutions. The method guarantees a smooth transition of the textures; From Figure 11(b), it can be seen that at a very close distance, the detail of the line texture boundary is almost no jagged phenomenon, and the texture aliasing is also reduced to the degree that the eyes cannot distinguish.

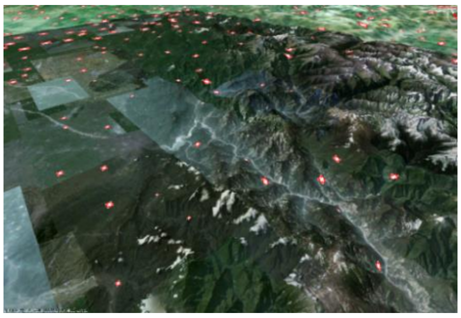


Figure 9. The fusion of hospital's point data

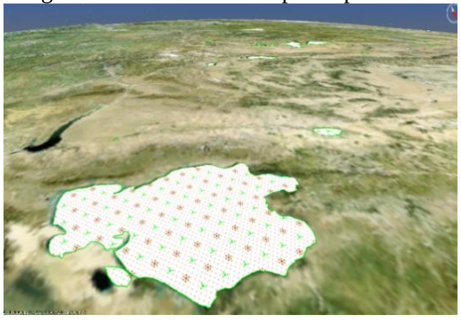
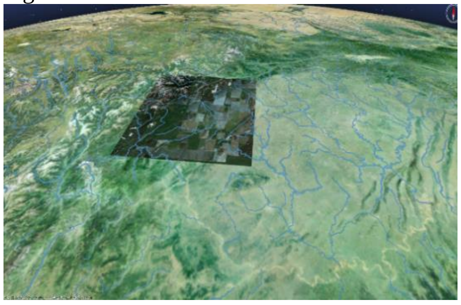
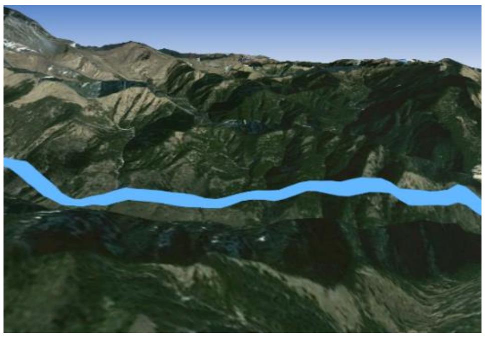


Figure 10. The fusion of meadow's surface data


(a) The fusion of longer distance


(b) The fusion of closer distance
Figure 11. The fusion of river's line data

To compare the performance of the method, we need to verify the real-time capabilities of the fusion and the efficiency of drawing the textures of vector objects. We select the national railway line data, the data size is 46.5MB, the vector object is 32830, the coordinate point is 265819, and a certain viewpoint in the scene is randomly selected. As shown in Figure 12, it can be seen that as the number of threads increases, the time taken to draw the textures continues to decrease. When the number of threads reaches 8 or more, the time required for drawing textures will be stable within a relatively small interval. At the same time, the rendering frame rate of the terrain scene will also steadily and slowly increase with the increase of texture threads, and it will be maintained at 180~200 frames.

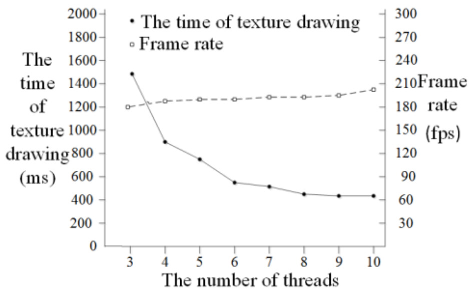


Figure 12. The trend with the number of threads

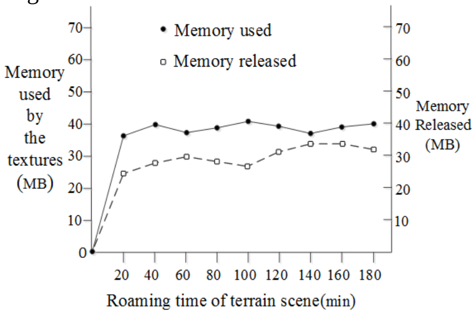


Figure 13. The trend of Computer memory

In addition, to verify the textures of vector objects released mechanism can avoid the textures growth caused by computer memory overflow. We integrate data from the national geographic range, including remote sensing imagery and elevations, seamlessly with county-level water system data with 129,181 vector objects. The rendering system will open up a piece of memory space, which can be used to calculate the change of vector objects’ textures memory size. Figure 13 shows the change of vector objects’ textures memory within three hours of random roaming the terrain scene. It can be seen that the computer memory occupied by vector objects’ textures is basically maintained between

30~50MB, and the released computer memory is maintained between 20~30MB. Therefore, the mechanism plays a significant role in preventing the rapid growth of computer memory.

4. CONCLUSION

This paper presents a viewpoint-related fusion method of massive vector data and 3D terrain based on multi-thread, which mainly includes the establishment of adaptive spatial indexing on vector data, the use of improved sub-pixel graphics engine AggExt generate vector objects’ textures, the vector objects’ textures released mechanism. Theoretical analysis and experimental results show that this method which adopts a multi-threading strategy, can ensure the rendering efficiency and real-time interactivity of the 3D scene while seamlessly blending the vector data with the undulating 3D terrain. The AggExt engine is combined with the LOD mechanism of the terrain so that it minimizes the aliasing caused by traditional methods and improves the quality of vector data fusion. In general, this method is more suitable for massive vector data and is more practical.

ACKNOWLEDGEMENTS (OPTIONAL)

This study was funded by the Basic Surveying and Mapping Project (A1705).

REFERENCES

Li, R., & Zheng, W. 2011. Real-time rendering high-quality vector data on 3d terrain. *Journal of Computer-Aided Design & Computer Graphics*, 23(7), pp. 1106-1114.

Kang, L. 2009. Terrain matching for three-dimensional visualization of two-dimensional gis vector data. *Computer Science*, 36(11), pp. 262-264.

Wang, J. J. 2013. Research progress on 3d visualization of vector data. *Guangdong Agricultural Sciences*.

Hong, C., Tang, X., Xie, Y., & Maoyin, A. S. 2010. Rendering vector data over 3d terrain with view-dependent perspective texture-mapping. *Journal of Computer-Aided Design & Computer Graphics*, 22(5), pp. 753-761.

Rui, X., & Zhang, Y. 2004. Overlaying vector data on 3D terrain. *Geoscience and Remote Sensing Symposium*, 2004. IGARSS '04. Proceedings. 2004 IEEE International (Vol.7, pp.4560-4563 vol.7). IEEE.

Wartell, Z., Kang, E., Wasilewski, T., Ribarsky, W., & Faust, N. 2003. Rendering vector data over global, multi-resolution 3D terrain. *Symposium on Data Visualisation* (pp.213-222). Eurographics Association.

Schneider, M., & Klein, R. 2007. Efficient and accurate rendering of vector data on virtual landscapes. *Journal of Wscg*, 15(15), 1--3.

Kersting, O., & Döllner, J. 2002. Interactive 3D visualization of vector data in GIS. *ACM International Symposium on Advances in Geographic Information Systems* (pp.107-112). ACM.

Schneider, M., Guthe, M., & Klein, R. 2005. Real-time Rendering of Complex Vector Data on 3d Terrain Models. *International Conference on Virtual Systems & Multimedia. Ghent*. pp. 573-582

Green, S. 2005. The opengl framebuffer object extension. http: download.nvidia.com

Wang, Y. 2003. An algorithm of self-adaptation layer-grid spatial index. Computer Engineering & Applications.

Zou, W., Fang, J. Y., & Liu, J. G. 2006. Research about visualization of hybrid multi-resolution terrain and vector data. *Journal of System Simulation*.

Cao, X. F., Wan, G., Feng, L. I., Ke, L. I., & Xiong, Z. M. 2013. Real time symbolization rendering method of vector map in 3d terrain environment. *Journal of System Simulation*.

Yang, L., Gong, A., & Jing, L. I. 2010. A model for massive 3d terrain simplification based on data block partition and quad-tree. *Acta Geodaetica Et Cartographica Sinica*, 39(4), pp.410-415.

Xue, Z. Y., & Wen, J. L. 2015. A method of processing the overlapping relations between annotation texts and the same color symbols based on gdi. *Engineering of Surveying & Mapping*.