www.cafetinnova.org

*Indexed in*
*Scopus Compendex and Geobase Elsevier,*
*Geo-Ref Information Services-USA, List B of Scientific Journals, Poland, Directory of Research Journals*

**ISSN 0974-5904, Volume 09, No. 04**

***International Journal of Earth Sciences and Engineering***

August 2016, P.P.1485-1493

# An Efficient and Dynamic 3D Visualization Simulation Method of Sea-level Rise Based on GPU

**CHEN QIYU[1], LIU GANG[1,2*], LI XINCHUAN[1, 2], ZHANG ZHITING[1, 2], WANG XIN[1] AND JIANG XIXI[1]**

[1]*School of Computer Science, China University of Geosciences, Wuhan 430074, China*

[2]*Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences, Wuhan 430074, China*

***Email:*** *chenqiyu403@163.com, *liugang67@163.com*

**Abstract:** Dynamic 3D visualization simulation is a significant technological method of sea-level monitoring and disaster prediction, with its efficiency and quality playing a direct and important role in the accuracy and timeliness for analyzing and researching. With the merits of two algorithms "Source Flood" and "No-source Flood", combined together and through a series of techniques and methods such as GPU parallel processing, Kriging surface interpolation, multi-constraints Delaunay triangulation, calculation of complex topological relation and spatial connectivity judgment, this paper proposes an efficient and dynamic 3D visualization simulation method. This method helps to distribute the workload of the complex computing and visual rendering with a certain strategy, avoiding the bottleneck of CPU-GPU Communication to some extent and enhancing the efficiency of dynamic simulation. Having been realized in the QuantyView3D geological visualization software platform, this technique is applied in the pilot project "Investigation of Geological Environment in Minjiang Estuary Area" launched by China Geological Survey.

***Keywords:*** *Graphics Processing Unit (GPU), Sea-level Rise, Dynamic Simulation, 3D Visualization*

## 1. Introduction

As the most potential natural disaster in the world, sea-level rise will have serious effects on global society, economy and ecological environment, etc (Bosello et al., 2007). Therefore, simulating and predicting influences caused by sea-level rise with a new quick, accurate and scientific method play a significant role in forecasting of marine disasters, dynamic simulation of disaster processes and disaster prevention and alleviation (Le Cozannet et al., 2014; Li et al., 2006).

Currently, the two main sea-level rise simulation methods are numerical simulation (Dawson et al., 2005; Valerio et al., 2003) and visualization simulation (Costabile and Macchione, 2015; Hou et al., 2012; Qu et al., 2009; Zhang et al., 2014). Numerical simulation, in most cases, quantitatively calculates numerical changes of some parameters after sea-level rise and its potential hazards, with results normally showed by words or charts, less intuitive. In contrast, with 3D visualization simulation technology, plenty of one-dimensional and two-dimensional monitoring data can be turned into 3D vector model data, showing visually the flood area, process and damages caused by sea-level rise, thus the variation trend of sea-level rise can be monitored, predicted and prevented.

Even specifically, two more methods are divided, namely "No-source Flood" and "Source Flood" algorithms (Izham et al., 2011; Kia et al., 2012; Sundara et al., 2015; Zhang et al., 2014). "No-source Flood" is an easy method with the principle of setting all the grids as flood when their altitudes are lower than the given water level at a particular moment. This method doesn't concern the spatial region connectivity judgment, resulting incapable of simulating some situations of complex landform such as mountain barrier. In practice, "Source Flood" is more common. Through spatial region connectivity judgment, it doesn't handle the spatial region units who cannot connect with source, thus realizing the simulation of the situation of complex mountain barrier.

With increasingly expanding of monitoring data and higher demanding of image precision, the conventional CPU-based rendering method, is disabled in real-time rendering and dynamic simulation in PC (Macedonia, 2003), so dynamic simulation efficiency has become a crucial problem of dynamic 3D visualization simulation. In recent years, with the development of graphic hardware technology, the new breed of Graphics Processing Unit (GPU) possesses a capacity of programming, enabling GPU to take partial computing, texture mapping and object rendering which are often performed by CPU conventionally. Through the coordination of CPU and GPU (Sugimoto et al., 2014), the efficiency of dynamic 3D visualization simulation is improved substantially.

According to the problems mentioned above, this paper applies a series of techniques and methods comprehensively such as Kriging surface interpolation (Cheng, 2013; Wang et al., 1999), Delaunay triangulation (Silveira and van Kreveld, 2009), calculation of complex topological relation and spatial connectivity judgment, improves "Source Flood" algorithm and proposes a method for efficient

and dynamic 3D visualization simulation of sea-level rise based on GPU. According to some distribution strategy, this method distributes the work of complex spatial vector computing and visual rendering to CPU and GPU, with computing load of CPU reduced and the whole simulation efficiency promoted.

## 2. CPU-GPU Distribution Strategy

CPU-GPU works together according to a determinate distribution strategy, promoting computing efficiency to some extent and improving simulation efficiency. Thus, the entire process of sea-level rise can be demonstrated dynamically with 3D visualization in real time.

### 2.1. Basic Mechanism of GPU Volume Rendering

Geometry is the basic unit of GPU visualization. Final rendering image results from the hardware's rasterization and corresponding disposal on geometry. Original volume model involves no geometry and the geometry is generalized only in the need of rendering, so they are called acting geometry accordingly (Rieder et al., 2011).

Traditional direct rendering algorithm needs to sample the whole 3D data sets when rendering, and hence it is very slow and it can't realize real-time rendering for regular scale Geometry Sets on PC (Blythe, 2006). Real-time volume rendering technology based on modern GPU achieves real-time rendering of large scale volume data exploiting flexible procedural model and 3D texturing ability of GPU.

In the stage of rendering, setting the render state is the initial step. Then model attribute value in memory needs to be turned into 3D texture data such as light intensity, color and opaqueness, which are stored in GPU texture cache. And triangular mesh index information of acting geometry is loaded into GPU index cache. According to index information, the vertex information in volume data corresponding to the place of vertex, normal vector and texture coordinate on geometry sections, are stored into GPU index cache. Consequently, the acting geometry can be rendered in the light of different shader program from the users setting, based on a certain sequence (Abellán and Tost, 2008; Hachaj and Ogiela, 2012; Rinaldi et al., 2012).

Geometry Shader allows GPU to generate dynamically and destroy geometric primitive data, expanding the function of GPU substantially. It can perform batch processing geometries according to vertex information, dispose functionally of vertex data output from vertex shader and generate new polygon vertex data in a high speed and large quantity (Kuo et al., 2007; Liu et al., 2012).

### 2.2. CPU-GPU Distribution Strategy in Dynamic Simulation of Sea-Level Rise

The given GPU-based dynamic 3D visualization simulation distribute the work of complex spatial vector computing and visualized rendering in the process of dynamic simulation to CPU and GPU, according to a certain distribution strategy (Figure 1). This method makes full use of flexible programmable technology and 3D texture ability of GPU, reducing the computing load of CPU and promoting the whole simulation efficiency.

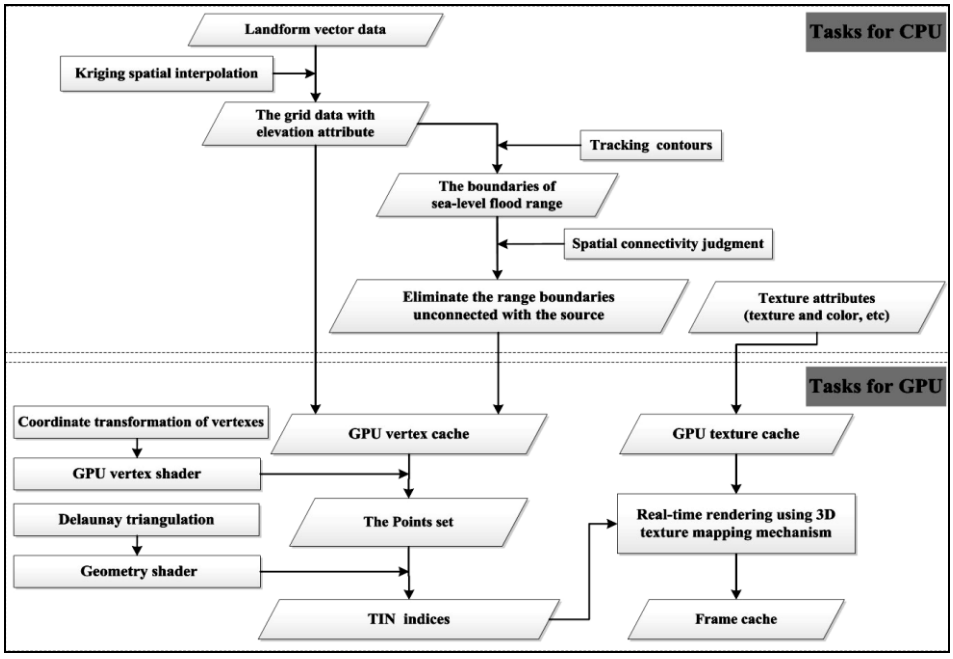


***Figure 1:*** *CPU-GPU distribution strategy in dynamic simulation of sea-level rise*

CPU-GPU Distribution Strategy (Figure 1) is the most important step in dynamic simulation of sea-level rise. CPU mainly deal with the disposal and transformation of spatial data, specific steps are: 1) Conduct Kriging surface interpolation on landform vector data (contour, elevation point, etc.), getting the grid data

with elevation attribute; 2) Regarding current sea-level rise elevation value $P_0$ as the target to track the contours with $P_0$ on the basis of grid data; 3) Make spatial connectivity judgment, and eliminate the boundary of sea-level flood range unconnected with the source (initiative direction of sea-level evolution); 4) Transfer the vector points set and texture attributes (texture and color) set by users to GPU to dispose. GPU achieves mainly the triangulation for the transferred spatial vector points set and real-time rendering by combining with 3D texture mapping mechanism, specific steps are: 1) Coordinate transformation of the points data in GPU vertex cache; 2) Designing Delaunay triangulation function with Geometry Shader and HLSL（High-Level Shading Language）and process the data output from GPU vertex shader to gain TIN (Triangular Irregular Network) indices; 3) Combine with spatial vector data and texture information in GPU texture cache and use 3D texture mapping mechanism for real-time rendering, getting a graphic of a particular moment, and then output it onto the display devices. Through the rotate operation above, dynamic 3D visualization simulation of sea-level rise can be obtained.

## 3. The Simulation Method of Sea-Level Rise

This paper proposes an efficient method of dynamic 3D visualization simulation of sea-level rise. The method is to allocate the simulation procedures to CPU-GPU to cooperate in virtue of a determinate distribution strategy. That is to say, data pre-processing and data transmission are done by CPU while triangulation of TIN mesh and 3D spatial rendering are disposed by GPU. To some extent, it avoids the bottleneck of CPU-GPU communication and enhances the efficiency of dynamic simulation. Specific steps are: 1) Landform data disposal based on Kriging surface interpolation and basic grid unit division; 2) Calculation of sea-level evolution range based on regular gird contour tracking; 3) Spatial lines connectivity judgment based on “Source Flood” algorithm.

The method of sea-level rise simulation adopted in this paper combines the respective advantages of “No-Source Flood” and “Source Flood” algorithms. Source is ignored in calculating sea level evolution range to reduce a large amount of computing time resulted from landform connectivity judgment in the procedure of contour tracking. After getting the range of “No-Source Flood”, this method is on the point of judging the spatial connectivity between the source (initiative direction of sea-level evolution) and the boundaries of flood range and removing the boundaries separated by landform. Compared to the traditional “Source Flood” algorithm using raster as the geographical unit, this method only needs to judge the connectivity between source and some boundaries. It reduces the amount of calculation and improves the simulation efficiency to some extent.

### 3.1. Landform Data Processing and Basic Mesh Generation

Landform data processing plays an important role on the precise calculation of sea-level evolution range. In the process of 3D visualization simulation, high-precision landform data (elevation point and contour, etc.) and remote sensing images are necessary. Process mass landform data with Kriging surface interpolation, and the regular surface mesh is generated and every grid nod gets an elevation attribute, all of which lay a solid foundation for the future work of contour tracking-based sea-level evolution range calculation.

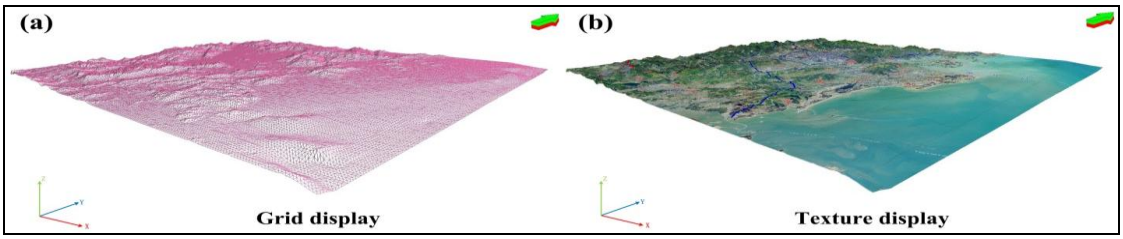


***Figure 2:*** *500*500 TIN mesh of surface: (a) the surface is displayed by grid and (b) the surface is displayed with texture image.*

Detailed calculation process runs as follows.

1) Initializing, users can set the number of rows and columns of grids and texture images (remote sensing images) as needed.
2) Gridding, generate m*n grid according to the number of rows and columns set above.
3) Based on Kriging’s spatial surface interpolation, take the landform data with elevation attribute (elevation point and contour, etc.) as samples and obtain elevation attribute of every grid nod in the spatial grids through Kriging interpolation.
4) Delaunay triangulation, use Geometry Shader to design Delaunay triangulation function and triangulate the spatial grid nods with elevation attribute to construct TIN mesh surface.
5) Texture mapping; map the texture images set above on the TIN mesh surface in terms of 3D texture mapping mechanism.
6) Rendering, 3D spatial visualization and TIN mesh surface displaying.

Step 1-3 above are performed on CPU to get the spatial grid nods with elevation attribute and texture images and transfer these data to GPU. Using Geometry Shader to design Delaunay triangulation function can operate the generation, texture mapping and rendering of mesh surface (Figure 2).

### 3.2. Calculation of Sea-Level Evolution Range

This paper integrates the respective advantages of "Source Flood" and "No-Source Flood" algorithms. In calculating sea-level evolution range, flood boundaries will be confirmed through track contours with flood height in this step. The source has not been taken into consideration in order to reduce the time of tracking contours and boundaries of landform barrier are eliminated by judging the connectivity between boundaries and the source.

The gridding points set has be acquired through disposing of Kriging interpolation. For the range consisted of $m \times n$ gridding points, the number of vertical line is $(m-1) \times n$ and the number of horizontal line is $(n-1) \times m$. The specific position of contour points on any side can be expressed as follows:

$$\begin{cases} xSide(i, j)(i = 1, 2, ..., m; j = 1, 2, ..., n-1), \\ ySide(i, j)(i = 1, 2, ..., m-1; j = 1, 2, ..., n). \end{cases} \quad (1)$$

$xSide(i, j)$ represents the contour points on the horizontal lines, and $ySide(i, j)$ represents those on the vertical lines.

In order to calculate the position of contour points on the grid boundary, first stage is to confirm the conditions of intersecting contour and grid boundary. If contour attribute value is $W$, the condition for the existing of contour points on boundary is that $W$ lies between the values of adjacent contour points.

If $W$ satisfies $(BB_{i,j} - W)(BB_{i,j+1} - W) < 0$, there exist contour points on the horizontal lines;

If $W$ satisfies $(BB_{i,j} - W)(BB_{i+1,j} - W) < 0$, there exist contour points on the vertical lines.

If this formula above is valid, the position of contour points can be figured out by the Linear Interpolation Algorithm and stored respectively in the arrays $xSide(i, j)$ and $ySide(i, j)$. These two arrays can be defined by

$$\begin{cases} xSide(i, j) = BB_{i,j} + \dfrac{W - BB_{i,j}}{BB_{i,j+1} - BB_{i,j}}, \\ ySide(i, j) = BB_{i,j} + \dfrac{W - BB_{i,j}}{BB_{i+1,j} - BB_{i,j}}. \end{cases} \quad (2)$$

In order to design tracking program, it is necessary to research the possible trends of a contour in the rectangular grid and establish the tracking conditions by confirming the relation between the trend of contour and the coordinates of contour points. As the contour points are located in the boundaries of grid, there are only four possibilities for contour to get through the adjacent grids: from bottom to top, from left to right, from top to bottom and from right to left.

The case of "from bottom to top" (Figure 3) shows that there are three possibilities of the positions of contour point a1 in Grid I: $xSide(i, j)$, $ySide(i, j)$ and $ySide(i, j+1)$, and one possibility of a2 in Grid II: $xSide(i+1, j)$. Obviously, through comparing the coordinates of a1 and a2, the rounded ordinate of point a1 is absolutely smaller than that of point a2. Therefore, "from bottom to top" can be operated if the condition $i_{a1} < i_{a2}$ is met. If there is a third point a3, it must be on other three sides of Grid II.

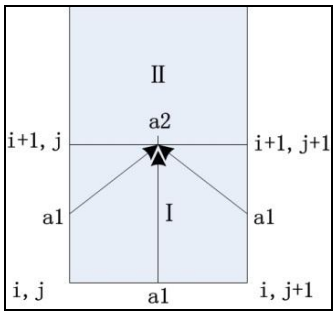


***Figure 3:*** *Tracking contour points from bottom to top: a1 is the contour point position in Grid I, and a2 is the contour point position in Grid II.*

Likewise, the tracking conditions of other three methods (from left to right, from top to bottom and from right to left) can be found (Table 1).

***Table 1:*** *Judgment method of four contour tracking conditions*

| | $bottom \rightarrow top$ | left $\rightarrow$ right | $top \rightarrow bottom$ | $right \rightarrow left$ |
|---|---|---|---|---|
| **Tracking Conditions** | $i_{a1} < i_{a2}$ | $j_{a1} < j_{a2}$ | $i_{a1} > i_{a2}$ | $j_{a1} > j_{a2}$ |

The contour can be divided into two types: non-closed contour and closed contour. The primary condition of tracking a contour is to find the starting point of this contour, and then to track it successively on all grid boundaries. As for the non-closed contour, starting point is on the rectangular outer boundary whereupon to track successively according to the rules above until leave the outer boundary. To track the closed contour must find the starting point on grid boundaries in the range. The contour point on any side

of inside the rectangular can be set as the starting point. And then it goes ahead by rule to be finished until starting point itself.

Tracking contour on regular grid based on Kriging interpolation can get the points set of sea-level evolution range.

### 3.3. Spatial Connectivity Judgment

Calculation of sea-level evolution range in the last step is a method of "No-Source Flood", without considering the influence of complex landform on sea-level rise. On the basis of the calculation of the flood range above, we can remove the range boundaries unconnected with the source by spatial connectivity judgment (Figure 4).

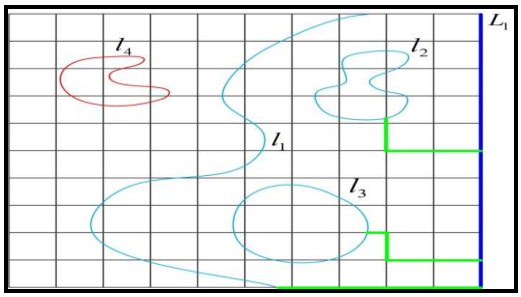


***Figure 4:*** *Spatial connectivity judgment of the boundaries of sea-level evolution range. The grid contains the points set with the elevation attribute gotten from Kriging interpolation. $L_1$ represents the flood source and $l_1$, $l_2$, $l_3$ and $l_4$ represent the flood range boundaries gotten from tracking the contour.*

Detailed computing steps of spatial connectivity judgment run as follows.

1) Get a boundary line (assume the elevation attribute of the line is $P_0$) of flood range and randomly select a point $O$ on the line;
2) Judge from the point selected, according to a certain priority direction (right, up, down and left). If the elevation attribute of the next grid point is smaller than $P_0$, this point is connected and then it won't be judged again by marking and adding it into the list of connected points set;
3) Loop operation of Step 2, and if some point is unconnected with that on other three directions, go back to the last nod to judge nods on other directions that unmarked;
4) If there is a way connected to flood source $L_1$, then this flood range boundary is connected with $L_1$ and thus getting out of looping. If returning to point $O$, this flood range boundary is separated by landform, then all the points on this boundary from the boundary points set of sea-level evolution range should be eliminated;
5) Process on all flood range boundary lines as shown above in proper sequence.

As shown in Figure 4, line $l_1$, $l_2$, and $l_3$ are connected with $L_1$. The green line is its connection line, which is not unique and is just an example in the figure. However, Line $l_4$ and $L_1$ are not connective so that $l_4$ should be removed from the list of sea-level flood range boundary. Compared with the regular method of "Source Flood", this method needs to judge the connectivity among the spatial lines with elevation value obtained from the last step. But it doesn't need to judge every grid, which saves calculation time to some extent and improves the simulation efficiency.

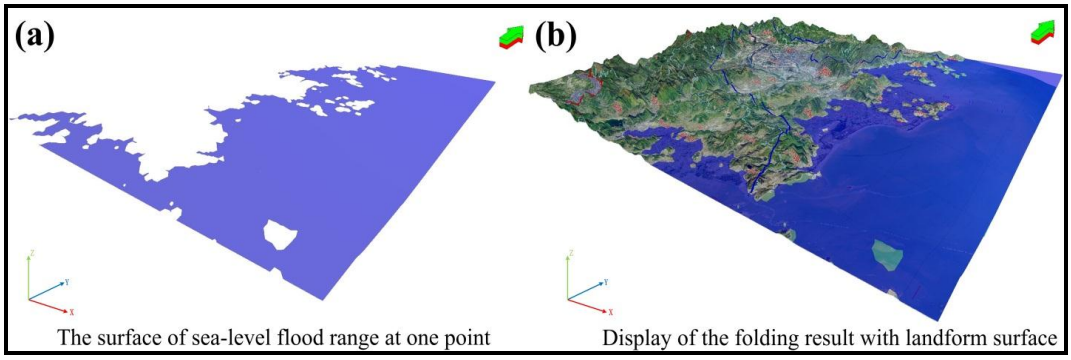


***Figure 5:*** *Sea-level flood range at one point*

Finally, this method transfers the spatial points set of complex topology to GPU and then uses Delaunay triangulation function designed by Geometry Shader to perform triangulation on the points set of inner and outer boundary line nods with complex topology, to generate TIN mesh surface and to finish triangulation and rendering of the complex surface (Figure 5).

### 4. Analysis of Simulation Results and Efficiency

This method has been applied in the pilot project "the Investigation of Geological Environment in Minjiang Estuary Area" of China Geological Survey. Minjiang Estuary covers Fuzhou City and its radiant economic zone. As the capital city of Fujian Province in eastern China, Fuzhou embraces complicated geological environment, being surrounded on three sides by mountains with sea on east, dominated by hills and tableland. According to the latest geological information, this area has seen four major seawater intrusions in history so that the whole Fuzhou Basin was ever flooded in the early of Holocene. In recent years, with global warming, sea level tends to raise so

much so that the economy, resources, life and property in this area are potentially affected. Therefore, it is of great significance to monitor the changes of coastline effectively, analyze their changing trends, as well as provide visualized and accurate forecast.

Compared with the common seed-fill algorithm considering source (Ding et al., 2004), this solving method based on the respective advantages of "No-Source Flood" and "Source Flood" in the first place is to gain the flood range boundary without thinking about the source through tracking contour, and then eliminate the range boundaries unconnected with the source by judging the spatial connectivity. At last we can get the "Source Flood" region, improving the efficiency and cutting down the computing time of CPU. From the GPU real-time rendering perspective, this method, in performing the flood surface triangulation, only concerns the points on the boundary and doesn't consider the projection point of landform grid within the outer boundary (Figure 6). The crucial factor affecting rendering rate depends on the division of landform mesh and the size of texture picture. The method proposed in this paper greatly reduces the number of triangulating triangles of flood region and improves the whole simulation efficiency. If the grid density increases, their contrast will be huger.

Experimental procedures have been accomplished under the circumstance of Microsoft Visual Studio 2010 by using C++ and OpenGL, with GPU programming performed by High-Level Shading Language (HLSL). Hardware configuration is Intel Core i5-2300 CPU and its basic frequency is 2.80 GHz, 4.0 GB RAM, NVIDIA GeForce GT 430 video card and the video memory is 1GB.

The input data are landform data (contour points and contour lines) of the study area and the high precision remote sensing image (Figure 7(a)). According to the number of rows and columns of landform surface grid and sea-level rise altitude set by users (Figure 7(b)), we can generalize the landform surface (Figure 7(c)), calculate the rising height of one frame and display the present rising height on dialog box in real time. Through the method in this paper of calculating, rendering and displaying, we set up a time animation of 3D dynamic visualization simulation of sea-level rise by controlling manually (Figure 7(d)) or calculating automatically (Figure 7(e)) and showing the sea-level flood mesh surface, in chronological order.

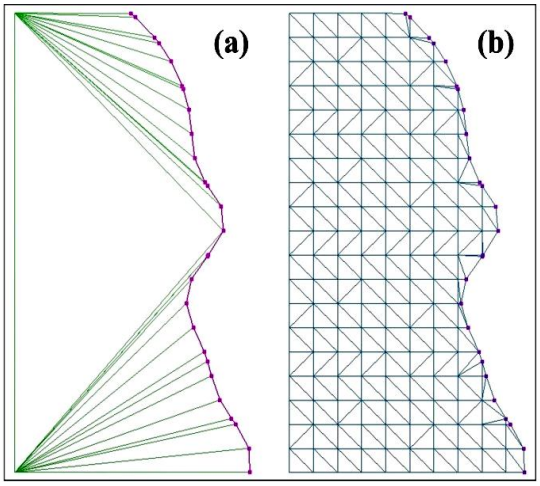


***Figure 6:*** *Triangulation comparison of flood surface: (a) is the subdivision result of this method with 27 triangles, (b) is that of traditional method with total 300 triangles.*

Figure 7 is a sample of mesh surfaces the grid of 500*500 with 500000 triangles totally in landform grid. The number of triangles of flood surface is not counted for it is not stable every moment. We can accomplish the dynamic simulation of sea-level rise through controlling manually and displaying automatically, show the current rising height in real time and even generalize animation video of the whole sea-level rise process. The real-time dynamic simulation can be achieved with such data available, processing fluent and dynamic visualization simulation efficient and high quality.

As shown in Figure 8, system rendering frame rate and utilization rate of CPU change in GPU accelerating method and CPU single calculation, with the improvement of grid density when we perform the 3D dynamic visualization simulation of sea-level rise based on this method. In the process of dynamic simulation, the rendering frame rate of GPU is seen changing gradually as grid precision improves in Figure 8(a). In GPU accelerating programming, grid density increases to 5000*5000 but the rendering frame rate maintains a high speed so as to meet the requirement of dynamic simulation and real-time rendering. However, when grid density increases to 2000*2000 in CPU single calculation, it is hard to realize real-time rendering. According to Figure 8(b), as grid density increases, the utilization rate of CPU without GPU accelerating method changes quickly while that in CPU-GPU coordinating work tends to vary gently. In virtue of part of computing, triangulation and all rendering tasks are handed over to GPU, and available CPU can operate other computing freely, increasing the efficiency of simulation, hence ensuring a large scale and high precision 3D dynamic visualization simulation of sea-level rise.

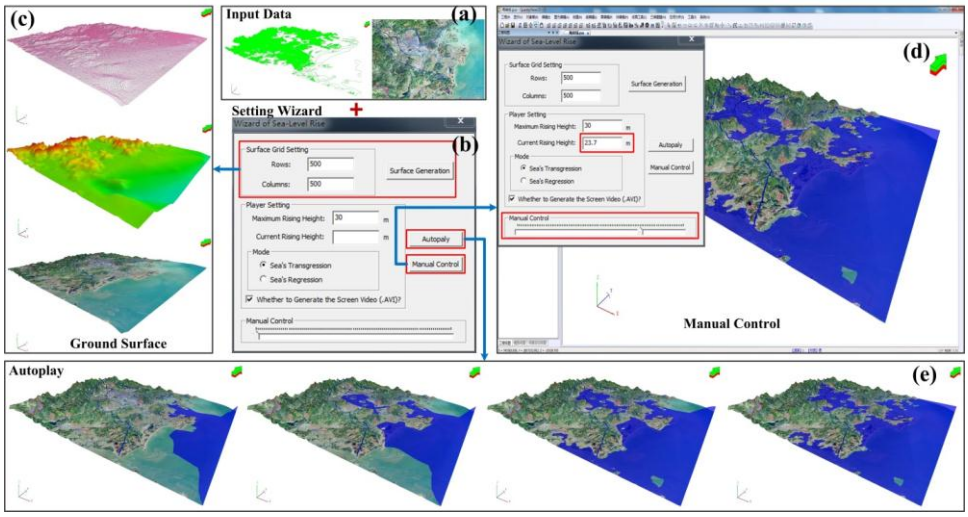


***Figure 7:*** *The whole process of sea-level rise: (a) Input data: landform data (contour points and contour lines) and remote sensing image; (b) setting and control wizard: set the number of rows and columns of landform surface grid and the highest altitude of sea-level, perform constantly the current height of sea-level rise in simulation process, select the simulation mode (transgression or regression), select whether to generalize video of simulation process, and select autoplay or manual control by ScrollBar; (c) landform surface (grid display, gradiant color display and texture display); (d) accomplish dynamic simulation of sea-level rise by ScrollBar manual control; (e) display automatically the dynamic simulation of sea-level rise at four different moments.*

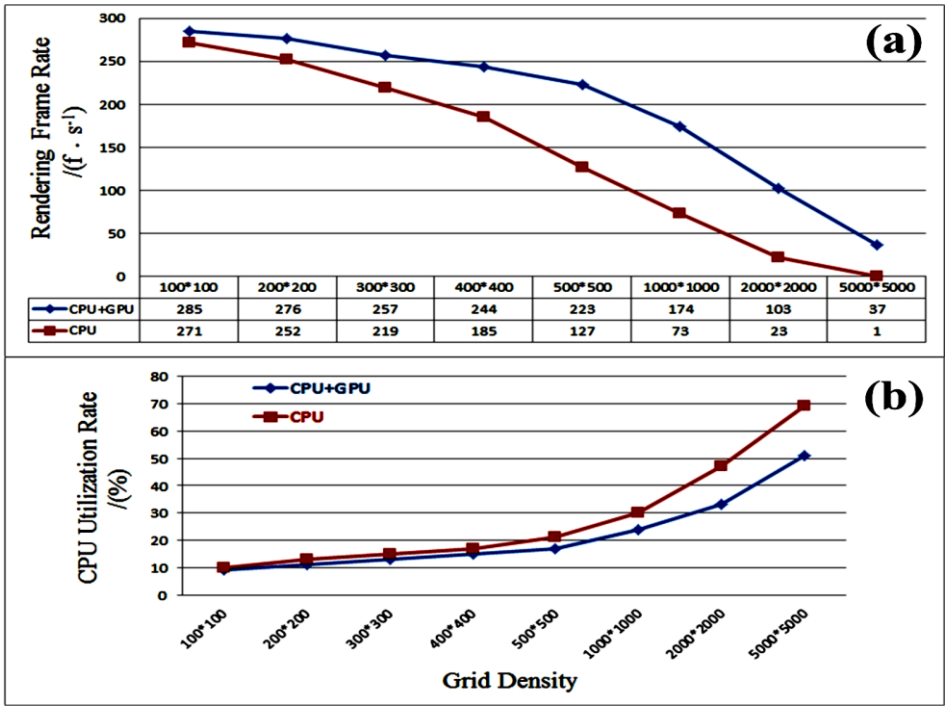


***Figure 8:*** *Comparison curves of rendering frame rate and utilization rate of CPU. (a) Comparison curves of rendering frame rate in CPU-GPU coordinating work and in CPU single calculation as grid density increases; (b) comparison curves of occupancy rate of CPU in the same case*

## 5. Conclusions and Future Work

This paper proposes an efficient method of 3D dynamic visualization simulation of GPU-based sea-level rise, which aims to perform the complex 3D spatial computing and visualized rendering task according to a certain allocation strategy. It distributes the complex spatial vector computing and visualized rendering tasks to CPU and GPU. By taking full advantage of the flexible programming technology and 3D texturing ability of GPU, this method eases the computing burden of CPU, avoids the bottleneck of CPU-GPU communication and even promotes the dynamic simulation efficiency. At the same time, the method combines the respective advantages of "Source Flood" and "No-Source Flood" algorithms and improves these two methods. It tracks contour to get the flood range without considering source and

then eliminates the boundary unconnected with source according to the spatial connectivity judgment. Eventually, the "Source Flood" range can be obtained, the efficiency of CPU to calculate the flood range improved. Triangulating the flood surface only needs to take into account the triangulation of points on the boundaries but it doesn't consider the projection point of landform grid within the outer boundary, which reduces the number of triangulating triangles of flood surface and eases the rendering burden of GPU. Through the combination of the parallel processing of GPU, Kriging interpolation, Delaunay triangulation, the calculation of complex topological relation and spatial connectivity judgment, this method promotes the efficiency and quality of dynamic 3D visualization simulation of sea-level rise, and guarantees the accuracy and timeliness. Apart from having been integrated into QuantyView3D geological 3D visualization software platform, this method also has been applied in the pilot project "the Investigation of Geological Environment in Minjiang Estuary Area" carried out by China Geological Survey.

However, other impact factors are also crucial in real flood process, such as artificial building and dyke-dam, etc. The future work is to analyze and judge more conditions, and then perform the 3D visualized real-time rendering on the dynamic effects of water flow in the progress of sea flood, exploiting the combined technologies of particle system and fluid simulation technology.

## 6. Acknowledgements

This work was supported by the "Investigation of Geological Environment in Minjiang Port Area" project cooperated by the Geological Survey of Fujian Province and China University of Geosciences, Wuhan, and partly by the Science Foundation of China (No. 41172300) and the National 863 Program of China (No. 2012AA121401). We are grateful to the editor in chief and the referees for their insightful comments and suggestions which led to substantial improvements in the manuscript.

## References

[1] Abellán P and Tost D. "Multimodal volume rendering with 3D textures", *Computers & Graphics*, Vol. 32, No.4, pp. 412-419, 2008.

[2] Blythe D. "The Direct 3D 10 system", *ACM Transactions on Graphics*, Vol. 25, No.3, pp. 724-734, 2006.

[3] Bosello F, Roson R and Tol R J. "Economy-wide Estimates of the Implications of Climate Change: Sea Level Rise", *Environmental and Resource Economics*, Vol. 37, No.3, pp. 549-571, 2007.

[4] Cheng T P. "Accelerating universal Kriging interpolation algorithm using CUDA-enabled GPU", *Computers & Geosciences*, Vol. 54, pp. 178-183, 2013.

[5] Costabile P and Macchione F. "Enhancing river model set-up for 2-D dynamic flood modelling", *Environmental Modelling & Software*, Vol. 67, pp. 89-107, 2015.

[6] Dawson R J, Hall J W, Bates P D and Nicholls R J. "Quantified analysis of the probability of flooding in the Thames Estuary under imaginable worst case sea-level rise scenarios", *International Journal of Water Resources Development*, Vol. 21, No.4, pp. 577-591, 2005.

[7] Ding Z X, Li J R and Li L. "Method for flood submergence analysis based on GIS grid model", *Journal of Hydraulic Engineering*, Vol. 6, pp. 56-60+67, 2004. (in Chinese with English abstract)

[8] Hachaj T and Ogiela M R. "Visualization of perfusion abnormalities with GPU-based volume rendering", *Computers & Graphics*, Vol. 36, No.3, pp. 163-169, 2012.

[9] Hou M L, Jiang X Z, Zhao X S and Xing H Q. "Analysis and simulation of sea level rise based on QTM", *Geography and GeoInformation Science*, Vol. 28, No.1, pp. 35-38+32, 2012. (in Chinese with English abstract)

[10] Izham M Y, Muhamad Uznir U, Alias A R, Ayob K and Wan Ruslan I. "Influence of georeference for saturated excess overland flow modelling using 3D volumetric soft geo-objects", *Computers & Geosciences*, Vol. 37, No.4, pp. 598-609, 2011.

[11] Kia M, Pirasteh S, Pradhan B, Mahmud A, Sulaiman W and Moradi A. "An artificial neural network model for flood simulation using GIS: Johor River Basin, Malaysia", *Environmental Earth Sciences*, Vol. 67, No.1, pp. 251-264, 2012.

[12] Kuo J, Bredthauer G R, Castellucci J B and von Ramm O T. "Interactive volume rendering of real-time three-dimensional ultrasound images", *IEEE Trans Ultrason Ferroelectr Freq Control*, Vol. 54, No.2, pp. 313-318, 2007.

[13] Le Cozannet G, Garcin M, Yates M, Idier D and Meyssignac B. "Approaches to evaluate the recent impacts of sea-level rise on shoreline changes", *Earth-Science Reviews*, Vol. 138, pp. 47-60, 2014.

[14] Li J L, Wang H Y, Zhang R S, Ge Y J, Qi D L and Zhang D F. "Disaster effects of sea level rise—A Case of Jiangsu coastal lowland", *Scientia Geographica Sinica*, Vol. 26, No.1, pp. 87-93, 2006. (In Chinese with English abstract)

[15] Liu B C, Bock A, Ropinski T, Nash M, Nielsen P and Wünsche B C. "GPU-accelerated direct volume rendering of finite element data sets", *Proceedings of the 27th Conference on Image and Vision Computing New Zealand*, ACM, Dunedin, New Zealand, pp. 109-114, 2012.

[16] Macedonia M. "The GPU Enters Computing's Mainstream", *IEEE Computer*, Vol. 36, No.10, pp. 106-108, 2003.

[17] Qu H, Cui X J and Dong W. Sea surface rise modeling and its implementation in the China Digital Ocean. *Marine Science Bulletin*, Vol. 28,

No.4, pp. 147-153, 2009. (in Chinese with English abstract)

[18] Rieder C, Palmer S, Link F and Hahn H K. "A shader framework for rapid prototyping of GPU-based volume rendering", *Proceedings of the 13th Eurographics / IEEE-VGTC conference on Visualization*, Eurographics Association, Bergen, Norway, pp. 1031-1040, 2011.

[19] Rinaldi P R, Dari E A, Vénere M J and Clausse A. "A Lattice-Boltzmann solver for 3D fluid simulation on GPU", *Simulation Modelling Practice and Theory*, Vol. 25, pp. 163-171, 2012.

[20] Silveira R I and van Kreveld M. "Towards a definition of higher order constrained Delaunay triangulations", *Computational Geometry*, Vol. 42, No.4, pp. 322-337, 2009.

[21] Sugimoto Y, Ino F and Hagihara K. "Improving cache locality for GPU-based volume rendering", *Parallel Computing*, Vol. 40, No.5-6, pp. 59-69, 2014.

[22] Sundara Kumar P, Praveen T V, Anjaneya Prasad M, Hari Priya B, Manogna P S L, Keerthi Harika A V. "Simulation of rainfall runoff using RS and GIS - A case study", *International Journal of Earth Sciences and Engineering*, Vol. 8, No.2, pp. 899-903, 2015.

[23] Valerio C, Alessandro V and Andrea Z. "Finite volume method for simulating extreme flood events in natural channels", *International Association for Hydraulic Research*, Delft, PAYS-BAS, 2003.

[24] Wang J B, Pan M and Zhang X D. "The Kriging interpolation method for scattered data poins", *Journal of Computer-aided Design & Computer Graphics*, Vol. 11, No.6, pp. 525-529, 1999. (In Chinese with English abstract)

[25] Zhang S H, Wang T W and Zhao B H. "Calculation and visualization of flood inundation based on a topographic triangle network", *Journal of Hydrology*, Vol. 509, pp. 406-415, 2014.