

Transmission and visualization of large geographical maps

Liqiang Zhang^{a,*}, Liang Zhang^b, Yingchao Ren^a, Zhifeng Guo^a

^a State Key Laboratory of Remote Sensing Science, Jointly Sponsored by Beijing Normal University and the Institute of Remote Sensing Applications of Chinese Academy of Sciences, Beijing 100875, China

^b School of Geomatics and Urban Information, Beijing University of Civil Engineering and Architecture, Beijing 100044, China

ARTICLE INFO

Article history:

Received 20 September 2007

Received in revised form

17 August 2010

Accepted 15 September 2010

Available online 8 October 2010

Keywords:

Large maps

Voronoi diagram

Out-of-core

Progressive transmission

ABSTRACT

Transmission and visualization of large geographical maps have become a challenging research issue in GIS applications. This paper presents an efficient and robust way to simplify large geographical maps using frame buffers and Voronoi diagrams. The topological relationships are kept during the simplification by removing the Voronoi diagram's self-overlapped regions. With the simplified vector maps, we establish different levels of detail (LOD) models of these maps. Then we introduce a client/server architecture which integrates our out-of-core algorithm, progressive transmission and rendering scheme based on computer graphics hardware. The architecture allows the viewers to view different regions interactively at different LODs on the network. Experimental results show that our proposed scheme provides an effective way for powerful transmission and manipulation of large maps.

© 2010 International Society for Photogrammetry and Remote Sensing, Inc. (ISPRS). Published by Elsevier B.V. All rights reserved.

1. Introduction

Geographical maps are one of the key data sources in the GIS community. They are represented as lines (arcs), polygons or their combinations. With the development of photogrammetry, remote sensing and related technologies, the large amount and multiresolution of acquired geographical data expanded quickly. The maps often contain a large number of vertices which make them difficult to be transmitted quickly on the Internet. An effective way to overcome this difficulty is to reduce the volume of the datasets and their complexity. For example, in the condition of satisfying a given precision, we firstly may transmit the simplified datasets to the clients, then send the remnant points to the clients and reconstruct maps locally. The topological relationships between one feature and other objects should be maintained consistently during the whole process.

In order to adapt the geographical data to computer capabilities like memory size, computing power, graphics support, and bandwidth, this paper presents an approach to construct a client/server application, which can transmit and render large vector map datasets dynamically. While maintaining the topological consistency of the vector maps, we simplify the vector maps efficiently and establish multiresolution models for vector maps. Our scheme allows the viewers to view different regions interactively at different levels of detail (LOD) on the network. This provides us with a great degree of control over the nature of the transmission.

2. Related work

Recently, many GIS software tools and algorithms are available to support real-time visualization and walk-through in three-dimensional (3D) environments. In this section, we give a historical overview of the research in this field.

McArthur et al. (2000) described an approach for generating a hierarchical, multiresolution polygonal database from raw elevation data using the wavelet transforms. But they failed to discuss the performance of the approach. Lindstrom and Pascucci (2002) presented a method for efficient out-of-core management and continuous adaptive terrain representation. The refinement framework is easy to implement to render large terrain datasets at high frame rates. A streaming hierarchical level-of-detail algorithm (HLOD) was proposed, which incorporates streaming into a HLOD rendering system that allows view-dependent refinement (Guthe and Klein, 2004). Cignoni et al. (2003) used a technique called P-BDAM for out-of-core management and interactive rendering of planet-sized terrain surfaces. Losasso and Hoppe (2004) introduced the geometry clipmap which can stream data to the graphic processing unit (GPU), showing clipmap levels and transition regions for a large terrain representation. Their GPU-accelerated method was based on a set of nested regular grids that centered the viewpoint. Geometry continuity was guaranteed by the transition regions between two grid levels using the GPU vertex shader. A compression algorithm was proposed to load the full terrain model in memory. However, on the one hand, this still required full CPU powers to compute vertex indices at each frame. On the other hand, their technique does not perform the feature-dependent mesh optimization, but uploads regular grids of different levels of mipmaps

* Corresponding author. Fax: +86 10 64853678.

E-mail address: zihaozhang2003@yahoo.com.cn (L. Zhang).

to the GPU. Based on the above work, Asirvatham and Hoppe (2005) further enhanced the method by performing nearly all computations on the GPU. The method is efficient, but it relies on shaders, which is impracticable, when targeting handheld devices.

Olsen et al. (2007) described a multiresolution method for curves and surfaces based on constraining wavelets. Then the constraint leads to an initial set of decomposition filters. The trial filters are then refined by an optimization step that seeks to reduce the error introduced during the decomposition stage. Mustafa et al. (2006) introduced an efficient Voronoi-diagram-based approach to perform dynamic simplification of large geographical maps. However, its precision is limited by the frame buffer's resolution and it is unable to avoid self-intersections. Guilbert and Saux (2008) described a global optimization technique based on a B -spline snake model to handle with the operators involved in the cartographic generalization process of lines. In order to avoid intersections or self-intersections, the consistency of the lines is checked and discrete operations such as segment removal are performed during the process. A limitation of the approach is the lack of topological information between lines. Rueda et al. (2008) presented a GPU-based rendering method of curved polygons defined by cubic Bézier curves. The method can direct render complex curved polygons in the GPUs. However, it often fails to handle curved polygons with self-intersections with explicit topological information.

In order to transmit large vector maps on the Internet, Yang et al. (2007) exploited the constrained removal operations of vertices to build continuous vector data on the server side, and consistent topology is maintained. Data reconstruction was implemented through a process of vertex addition on the client devices. Huang et al. (2001) presented a framework for interactive web-based visualization by using Java 3D-based hybrid Internet computation through the simulation steering technique. However, the framework is hard to handle a large number of datasets. Pajarola and Widmayer (2001) described a virtual reality (VR) prototype system. A client runs the VR component interacting via the network with a server that runs an object-oriented database containing geographic data. They integrated a geometric index into the database, and proposed the storage and retrieval schemes for location-oriented and LOD-driven dynamic loading. Coors (2003) proposed a data model for 3D geometry and topology in a 3D-GIS. Together with a spatial index method called P -tree, progressive transmission was used to visualize 3D-GIS data in the networking environments. El-Sana and Sokolovsky (2003) developed a web-based 3D scene system. In order to improve adaptive visualization and reduce transmission time, the system uses the cache and a prediction mechanism on the client side. They subdivide the view-dependent tree into blocks which allow selective refinement and maintain on the server side a list of active nodes for each client connected. The server responds the requests from the clients for sending the update operations needed to satisfy the visual query. Kim et al. (2004) proposed a client/server framework for view-dependent streaming of progressive meshes: the servers send the base mesh and the vertex hierarchy of 3D models to the client. The client creates further requests for selective refinement operations, and the server continues to send the rest level of detail of the models until the requests stop. However, the application cannot support interactive exploration of models, which involves selective refinement. Danovaro et al. (2006) designed a client/server system for allowing an application to deliver a mesh progressively, which overcomes some limitations of previous work. The client can make simple computation and it can store more than the mesh generated for the visualization process. It builds a partial copy of the MT model as it receives information from the server. An open source project called GRIFINOR which allows communication with object database over the network (www.grifinor.net). GRIFINOR applies a client/server architecture. Nevertheless the design is such that

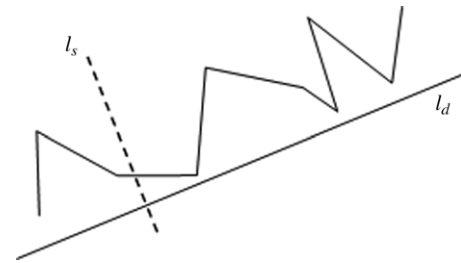


Fig. 1. A monotone polyline.

both client and server components are tightly coupled. This means that a GRIFINOR instance can behave as both client and server, which allows to build a peer-to-peer network. However, a specification of such network is currently missing (Kolar, 2006).

3. Simplification of vector maps

This study mainly focuses on polyline simplification since the majority of map features are represented as lines or polygons made up of lines. For large maps, it is very difficult to read into main memory once, so we should segment the whole map into small tiles, and handle these tiles in turn. During processing each tile, the entire simplification procedure is summarized in the following five steps:

- (1) Split original polylines into *monotone* sub-polylines.
- (2) Generate *segments* for each sub-polyline.
- (3) Store the vertices of *non-compliant* segments and remnant segments respectively.
- (4) Link remnant segments to generate the simplified sub-polylines.
- (5) Connect the simplified chains to create the final results.

Among the above five procedures, the steps from (1) to (3) are implemented on the server at the preprocessing, and the steps (4) and (5) are performed on the client.

3.1. Generalization of monotone sub-polylines

To avoid invalid topologies, we first split the polyline and build *monotone* pieces. A polyline l is *monotone* if there is a line l_d , for any perpendicular line l_s of l_d , l_s has at most one intersection with l (see Fig. 1). It is evident that the simplified monotone polyline is still monotone. One important property of a monotone polyline is that it does not generate self-intersections.

To get monotone pieces, we connect each pair of successive vertices to form vectors. Then we define the angle formed by two vectors \vec{p} and \vec{q} as follows: if the smaller angle formed by \vec{p} and \vec{q} is determined by counter-clockwise rotation from \vec{p} to \vec{q} , it is the correct angle; otherwise, the angle is treated as negative. Therefore, the angle falls in $(-180^\circ, 180^\circ]$. For a polyline $l = (v_0, v_1, \dots, v_n)$, the *edge-angle* θ_i associated with the edge $v_i v_{i+1}$ is defined by the angle from $\vec{v_0 v_1}$ to $\vec{v_i v_{i+1}}$. Obviously θ_0 equals zero. Chandru has proved that l is monotone if and only if its edge-angle sequence $\{\theta_0, \theta_1, \dots, \theta_{n-1}\}$ satisfies $(\theta_{\max} - \theta_{\min}) < 180$, where $\theta_{\max} = \max(\theta_0, \theta_1, \dots, \theta_{n-1})$ and $\theta_{\min} = \min(\theta_0, \theta_1, \dots, \theta_{n-1})$ (Chandru et al., 1992). Therefore, starting with θ_0 , we compute the edge angles in sequence and update θ_{\max} and θ_{\min} simultaneously. If $(\theta_{\max} - \theta_{\min}) \geq 180^\circ$ at θ_i , split the polyline at v_i . Then we further start with v_i and repeat the process. In this way, we can generate monotone sub-polylines with time complexity $O(n)$.

Through the above process, an original polyline has been split into monotone sub-polylines, so that the simplified results do not self-intersect. The next step is to simplify the monotone sub-polylines and prevent the simplified results from intersecting with each other.

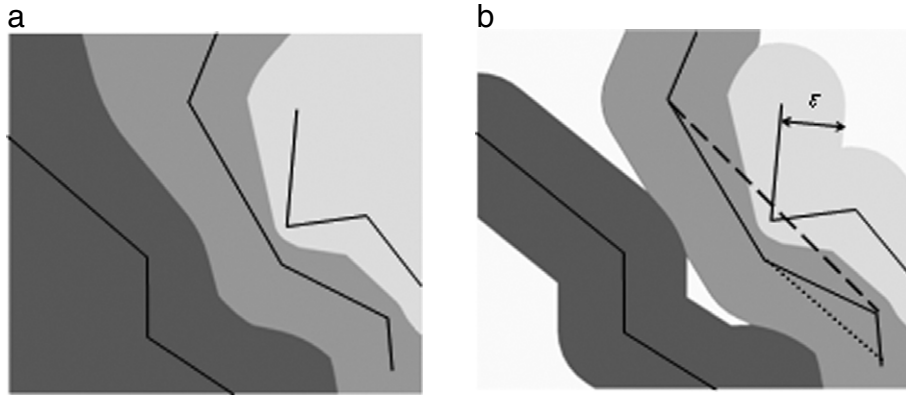


Fig. 2. (a) Voronoi diagrams of three polylines. (b) ε -Voronoi cells, the non-compliant segment (dashed line) and the compliant segment (dot line).

3.2. Simplification of the sub-polylines

ε -Voronoi diagrams are applied to constrain the simplification of monotone sub-polylines. The aim is to avoid the intersections. Fig. 2(a) illustrates the Voronoi diagrams of three polylines. A primitive's ε -Voronoi cell is the portion of its Voronoi cell that lies within ε of the primitive (see Fig. 2(b)).

To avoid the intersections, Mustafa et al. (2006) proposed a polyline simplification algorithm that removes *non-compliant segments* by ε -Voronoi cell. A polyline's *segment* is a line segment connecting any two vertices of this polyline. If a segment is *compliant*, it completely lies within the polyline's ε -Voronoi cell; otherwise, it is a *non-compliant* segment (see Fig. 2(b)). It is obvious that the compliant segments of a polyline do not intersect with the compliant segments of other polylines. On the contrary, non-compliant segments may cause intersections; consequently, removing all the non-compliant segments can avoid intersections. However, the above method may create self-intersections. In this study, we improve the method to avoid both intersections and self-intersections by computing ε -Voronoi cells for each monotone sub-polyline rather than the polylines. The major simplification steps are as follows:

- (1) A set of shortcut segments for each monotone sub-polyline is generated.
- (2) Render the ε -Voronoi cell for each monotone sub-polyline in the stencil buffer. Each monotone piece's Voronoi region is drawn with a unique stencil value.
- (3) Draw each shortcut segment in the color buffer with a unique color with the stencil test enabled.

For l_i the stencil test is set to pass only if the stencil value is not equal to i . Because the stencil value of the ε -Voronoi cell is i , the stencil test fails when its segment completely in the Voronoi region is rendered. Thus, compliant segments cannot be drawn in the color buffer; otherwise, the colors of non-compliant segments appear in the color buffer.

- (4) We scan the color buffer; if the color of a shortcut segment is found, the shortcut segment is non-compliant and should be removed. Last, the remaining compliant segments are connected to get the final result.

With the help of the depth and stencil buffers, the Voronoi regions are generated quickly without complex geometric computation. Since polylines are represented by pixels at the color buffer, a non-compliant segment may be occluded by other segments in the color buffer, so it cannot be identified and removed. Therefore, it is necessary to repeat the processes for clearing the color buffer, rendering the segments, and eliminating non-compliant segments until no new non-compliant segments are found. Generating fewer shortcut segments can reduce the iterations.

Douglas–Peucker algorithm (Douglas and Peucker, 1973) is used to select an optimal set of shortcut segments. The original polyline's starting and ending vertices are connected to get a segment. Calculate the distances from the intermediate vertices to it, and find the vertex with the maximum distance. If the distance is less than ε , we add the segment to the segments set. Connect the vertex having the maximum distance with the starting and ending vertices respectively to generate two segments. Repeat this process until no more segments are generated.

For a monotone sub-polyline, its compliant segments completely lie in its ε -Voronoi cell. So the simplified sub-polyline cannot cross with other simplified polylines. Because the simplified sub-polylines are monotone, they will never cross themselves. Thus all the intersections and self-intersections are removed. Fig. 3 illustrates the main procedure of the proposed method.

4. Framework of transmission and reconstruction

Generally, implementation of our application is divided into two procedures: one is vector map refinement, which includes the multiresolution hierarchy traversal on the server and network transmission, and the other is the rendering process, which involves the scene refresh and the actual scene visualization on the client.

4.1. Overview

To accomplish the goal for transmission and rendering of massive geographical maps, our approach combines view-dependent LOD simplification and out-of-core rendering algorithm. Fig. 4 illustrates the architecture of our scheme. The architecture is composed of three components: the client, the server and the database/files.

During preprocessing, the geographical maps and underlying terrains are subdivided into titles. Each part is decomposed into a set of clusters. The clusters are passed to a cluster hierarchy generation algorithm. Each title of the maps is simplified and generated LOD models using the method introduced in Section 3. These models are stored in the binary tree. The hierarchy builds a balanced hierarchy in a front-to-back, top-down manners. The database/files store the multiresolution map models, and the server traverses the hierarchy to select map updates for the client. The client is responsible for reconstructing the progressive data transferred from the server and rendering them dynamically.

When a request is issued, the client sends the requests to the server for multiresolution models falling inside the view frustum. The server delivers the base models of the spatial objects to the client first. Then, the client interacts with the users and sends the view parameters to the server for fine models update. The



Fig. 3. Polyline simplification procedure. (a) Original polylines. (b) ϵ -Voronoi diagrams of monotone sub-polylines. (c) Ultimate simplified results (bold black lines).

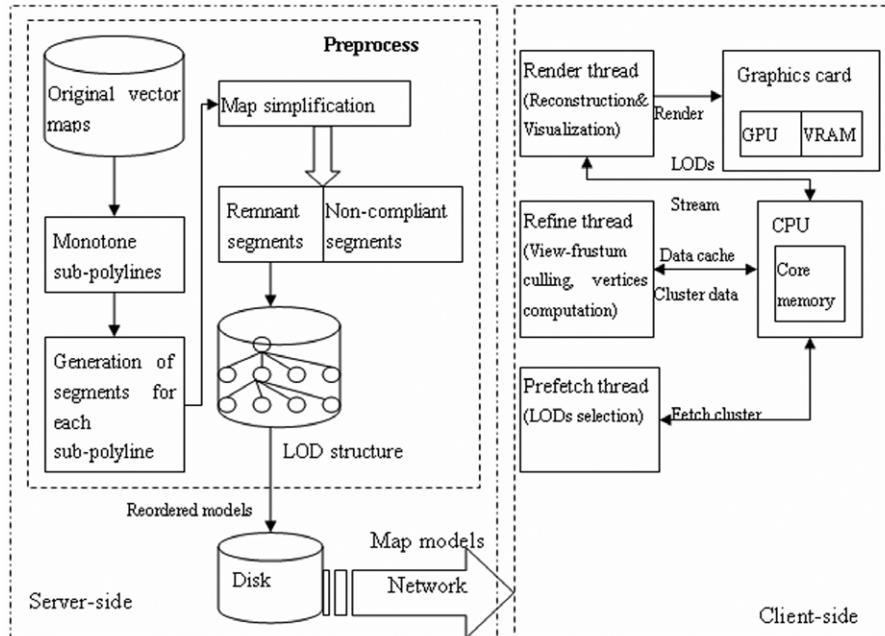


Fig. 4. The architecture of the application.

simplification process of the server produces a stream of map simplification operators. These update operators are sent to the client and the selective model is reconstructed and rendered. As the viewers navigate closer to the interested region, the details will be updated and refined accordingly. After all the datasets reach the client or the added details become imperceptible to the viewers, the process stops.

4.2. Progressive transmission

The server is responsible for generating LOD map models offline and creating XML metadata files for the complete collection of datasets. The metadata files describe the information such as projection, size, location, URLs and so on. The URLs mainly present the location of the spatial datasets on the server, and remote users can access the datasets from the URLs. The metadata file is a useful mechanism for organizing collections of the related features. The metadata files and LOD map models are placed on the location accessible to the Web server for later transmission to the client.

The XML metadata file is first transmitted to the client through a file transfer protocol. The client reads the metadata file and determines the current positions of the datasets from the file. If the datasets are on the client, the render operators on the client read the data from the cache or the hard disks. Otherwise, the client sends a request to the server for the LOD models.

After receiving the request from a client, the server acquires the proper datasets from the database. The coarsest map M_0 is transferred to the client firstly, and then the server transfers the remained vertices as the binary stems using the following data structure.

```
struct Vertex {
    double x, y;
    long ObjectsID;
    long Position;
}
```

where x and y store the coordinates of the vertex. *ObjectsID* is the identifier of the element that the vertex belongs to. *Position* records the position of the vertex in the vector feature. After finishing the simplification, the attributes of the vector data are also transferred to the client.

4.3. Out-of-core map reconstruction

The client initially receives the coarse models from the server. By monitoring the process, the client can identify the portion of the maps currently being examined. The client renders the maps based on the received subsets of the large collection. If needed, the client requests more detailed data from the server. It incorporates them into the data array as they arrive, progressively improves the accuracy of the rendered maps. The downloaded datasets are cached into memory and ordered in a priority queue. The regions recently viewed have higher priority in the queue, whereas those not recently viewed have lower priority. The relative priority can be changed as the viewpoint moves. Once the viewpoint changes, message priorities will change and additional data might be required. The LOD models allow the system to prioritize parts of the maps that have the biggest visual quality.

On the client, our out-of-core rendering algorithm uses the paging mechanism in the operating system (OS) by mapping a

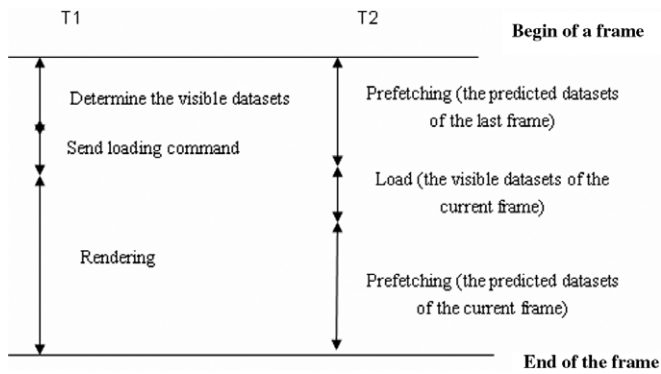


Fig. 5. Parallel of rendering and I/O. The process T1 is responsible for rendering maps, and T2 for I/O operations.

file into read-only logical address space (Lindstrom and Pascucci, 2002). We choose the OS's virtual memory management, because it can effectively optimize the disk access patterns and perform efficient memory management, which simplifies the design of our out-of-core algorithm. With our out-of-core algorithm, data need not completely reside in main memory all the time but is loaded from the hard disk on demand. By using spatial data structures and efficient spatial queries, we can quickly determine the potential visibility of large subsets of all vector map models, even before they are loaded into main memory. The entire LOD hierarchy is in main memory and we fetch the proper level-of-detail map models needed for the current frame as well as prefetch some other LOD models for subsequent frames. Additionally, we store the vertices of vector maps in GPU memory. By rendering the primitives directly from the GPU, the rendering performance is improved.

To compensate for network latencies, our out-of-core rendering algorithm supports multi-threads including data prediction, refinement and rendering run in the back end of the application communicating with the external processes, so that multiple tiles may be requested at once. Each thread opens a connection back to the server for requesting the maps. The prediction thread mainly computes the next frame to be rendered. This thread will be triggered when the computer is idle. It keeps as many leaf nodes as possible in main memory, starting at leaves with the highest priority. Lower-priority vertices that have stayed in memory the longest are removed from memory first, so that only data recently rendered or still within the extended view frustum remains in memory. The refinement operation thread is invoked to compute the visible region once the viewpoint changes, responsible for vertex array coordinates update. The render thread implements the multiresolution rendering. Since the hard disk is the slowest component of the application and new geometries need to be loaded from it, to reduce latency, the prefetch that runs parallel to the rendering thread is used. As a result, the graphics card has much less work to do and can focus on rendering the maps (Fig. 5).

High throughput from graphics cards can further be achieved by storing the vertices of vector maps on the GPU at render time, thereby reducing the data transferred from the CPU to the GPU during each frame. A second caching mechanism loads the visible parts into the graphics card's video RAM. As a result, we only transmit the vertices of these models to the GPU and the other vertices are cached in the GPU memory.

4.4. Precision retaining

When drawn at large scales in the frame buffer, a vector feature may be represented by several pixels or a single pixel, so the whole vector map is subdivided to reduce inexactness and occlusions caused by the rasterization (Yang et al., 2010). The rule of this

division is to guarantee that each line segment can be represented by at least two pixels. The map should be divided into at least n_W sub-regions in the width direction: $n_W = \lceil W/(d \bullet w) \rceil$, where d is the average distance between two adjacent vertices of a polyline, and d can be achieved by roughly sampling or visual estimation; w is the screen width; W is the map width. The $\lceil \cdot \rceil$ function rounds a number up to an integer value. Otherwise, different vertices in the map may cover the same pixel in the frame buffer. We then divide the map into n_h sub-regions in the height direction.

For most computer graphics cards, the stencil value ranges from 0 to 255. If the count of monotone sub-polylines is larger than 255, we use simple heuristics to color the adjacent ϵ -Voronoi cells with different stencil values. For each sub-polyline, before rendering the ϵ -Voronoi cell, we scan its surrounding pixels in the stencil buffer and record stencil values of these pixels. Then this sub-line's stencil is set different than any of these pixels' stencil values.

5. Experiments

5.1. Our application

Our application for data transmission and visualization has been developed in C++ and OpenGL, and integrated into COM components (Zhang et al., 2005). The COM components are composed of three modules. The first one is the *http* client being responsible for downloading the data from the server. The second one is the scene manager which loads the data into the data repository and generates the multiresolution 3D scene. The third one is the 3D analysis module handling geographic analysis, such as the shortest route analysis, hydrological analysis and more. These modules are interconnected, providing an integrated networking environment for large geographic data rendering.

We have validated the performance of our method. The server side is an IIS/Windows server. The experiment was undertaken on the Internet using a 512 K modem connection.

5.2. Performance analysis

Three computers as the client machines simultaneously visit the server. The first client side used is an Inter Core2 at 2.4 GHz with 4 GB of RAM and an NVIDIA GeForce 8800 GTX (called *computer-I*). The second is a P4 2.8 GHz processor with 1G and NVIDIA FX3400 3D graphic card (called *computer-II*). The third client side is Processor Intel(R) Core™2.4 GHz 2 CPU, 2G RAM, and ATI Radeon HD 4650 (called *computer-III*).

We used two approaches to visualize the dataset list in Table 1 on the network. One is the method proposed in this paper (called *M-I*). The other is the method without using progressive transmission and progressive reconstruction (called *M-II*).

The dataset I-3 and the corresponding terrain models were subdivided into tiles. Each terrain tile has 512×512 height points and each map tile has the same size as the terrain.

Table 2 gives the time for transferring and rendering the three datasets using the two methods. From the table, we see that the method *M-I* can greatly reduce the time of the spatial data transferred on the network compared with *M-II*. The reason is that the speed of the progressive simplification and reconstruction is faster than that of the transmission on the network for the reduced data volumes. By progressive transmission of the vertices, our approach significantly reduces the bandwidth requirements. Processing and rendering functionality on the recent GPUs is exploited, so that our application achieves high frame rates.

We also compared our method against the one presented by Mustafa et al. (2006) using the datasets described in Table 3. From the results in Table 4, we see that these two methods cost nearly the same amount of time to get the same compression ratios.

Table 1
The experimental vector datasets.

Dataset type	Data volume (kb)	No. of polylines/polygons	No. of vertices
I-1 Beijing transport map	13,434	12,651	814,023
I-2 Hawaii soil map	16,205	5,548	987,204
I-3 The whole China 1:1000,000 resolution soil map	227,262	254,303	44,092,832

Table 2
Statistical information about the data visualization.

Computer ID	Original maps	Simplification time (min)	Transferring time (s)	Rendering in full resolution time (min)
I (with M-I)		2.31	6.98	3.64
I (with M-II)	I-1		32.37	5.15
II (with M-I)		2.73	8.42	4.96
II (with M-II)	I-2		36.25	6.87
III (with M-I)		51.52	398.12	45.46
III (with M-II)	I-3		451.71	81.58

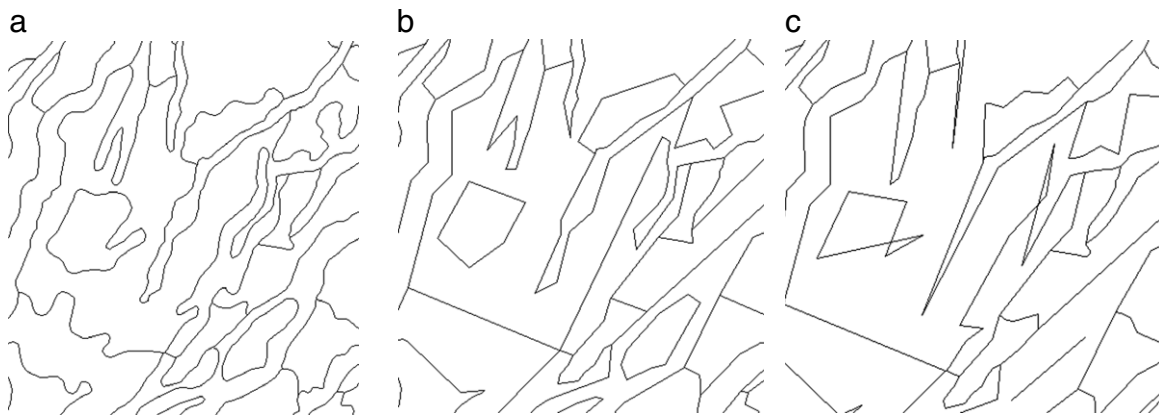


Fig. 6. Example of the map II-4 simplifications generated by the two methods. (a) A small region in the original map which has 987,204 vertices. (b) A small region in the simplified map generated by the proposed method with 70,630 remaining vertices. (c) The same region in the simplified map generated by Mustafa et al. (2006) with 70,461 remaining vertices.

Table 3
Tested vector datasets.

Dataset type	No. of polylines/polygons	No. of vertices	Source
II-1 Land use/land cover (LULC) map	1197	90,145	Hawaii island, USA http://www.state.hi.us/dbedt/gis/lulc.htm
II-2 500-ft contour map	305	93,187	Hawaii Island, USA http://hawaii.gov/dbedt/gis/cntrs500.htm
II-3 Stream map	2651	114,023	Hawaii island, USA http://hawaii.gov/dbedt/gis/huntareas.htm
II-4 Soil map	5548	987,204	Hawaii island, USA http://hawaii.gov/dbedt/gis/soils.htm

The proposed method generates fewer segments, and thus spends less time in detecting the non-compliant segments. On the other hand, our method computes ϵ -Voronoi cells for the monotone sub-polylines instead of the original polylines. The main advantage is that it avoids both intersections and self-intersections, whereas the method of Mustafa et al. may introduce self-intersections. Fig. 6 shows the simplifications from both methods; ESRI ArcMap software was used to check the topological relationships in the proposed simplified maps, and no topological error was found. With almost the same compression ratios, we see that the proposed method can keep the topological consistency exactly.

The proposed method was also compared with the geometric methods. The method presented by Yang et al. (2007) was chosen for comparison, because the latter can prevent the intersections and self-intersections. To reduce computing time, Yang et al. removed small features before simplification. In contrast, our approach has kept all the features. In the experiment, Yang's method is re-implemented without removing small features. The results are shown in Table 4. When the data volume is small, our method does not have obvious advantages compared with Yang's method. The reason is that clearing and reading the frame buffer takes a certain period of time. However, the method proposed in

Table 4
Comparisons of our algorithm with the methods by Mustafa et al. (2006) and Yang et al. (2007).

Original maps	Our method		Yang et al.		Mustafa et al.	
	RV	ST (s)	RV	ST (s)	RV	ST (s)
II-1	9,530	4.2	9,644	2.7	9,294	3.7
II-2	1,831	3.5	1,806	3.4	1,862	2.7
II-3	5,720	4.5	5,693	3.2	5,835	3.2
II-4	70,632	24.8	70,648	45.1	70,461	27.1

Note: RV = remaining vertices, and ST = simplification time in second.

this study is more efficient in simplifying large datasets. It avoids complex geometric computing process and takes advantage of modern graphics hardware. Therefore, our method is able to speed up the simplification process.

The details of the different LOD maps are controlled by ϵ . We can build map models with different feature details through giving the values of ϵ . The following three figures (from Figs. 7 to 9) illustrate different levels of the dataset I-3 rendered in the networking environments. At first, the server sends the coarse model (see Fig. 7) of the I-3 to the client. Once the coarse model has been rendered, the client requests the additional dataset from the

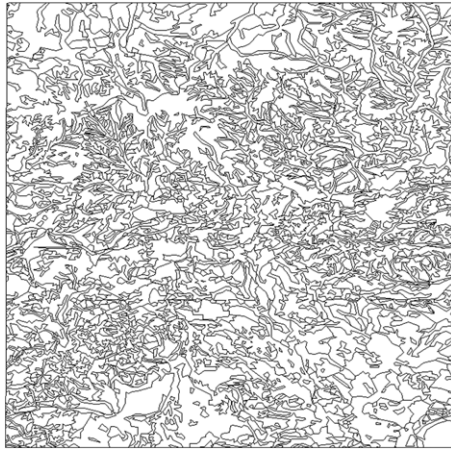


Fig. 7. The transmitted simplification map of the dataset I-3, $\varepsilon = 0.2\%$.

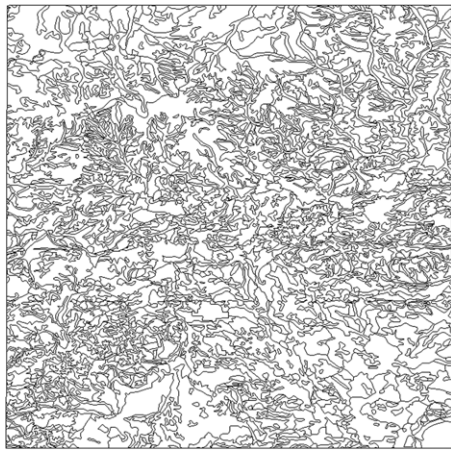


Fig. 8. The transmitted simplification map of I-3, $\varepsilon = 0.05\%$.

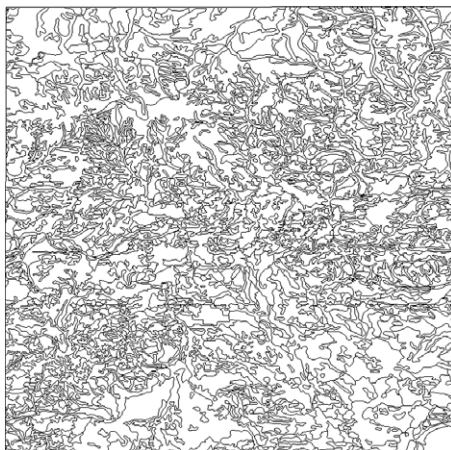


Fig. 9. The reconstructed original dataset I-3.

server. The new vertices are inserted into the coarse map model in the reverse order of the removal sequence on the client. The details of the map become fine accordingly (see Fig. 8). Until all vertices of the map are transferred to the client, the transmission process is finished. After receiving all the datasets, the client implements the reconstruction (see Fig. 9).

From these figures, we find that the maps with different details on the client can keep the topological consistency with the original maps.

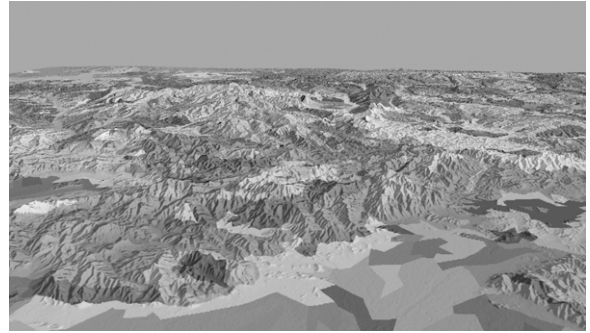


Fig. 10. The view-dependent reconstructed LOD models of the dataset I-3 overlay ($\varepsilon = 0.5\%$).

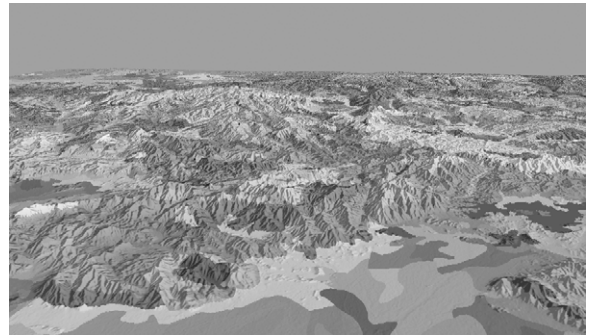


Fig. 11. Overlay of the reconstructed dataset I-3 on the client.

In order to further validate the performance of our scheme for transmitting and rendering vector maps, we overlay the transmitted map features on the corresponding terrains on the client. Fig. 10 shows the dataset I-3 is overlaid on the corresponding SRTM 90 m Digital Elevation Data. With the progressive transmission, the map becomes finer. The transmission process is stopped until the original map is restored on the client side. Fig. 11 illustrates the original dataset I-3 is rendered on the same terrains as that of Fig. 10. As shown in Figs. 10 and 11, we cannot find aliasing artifacts or gaps. Moreover, the visual quality of the image in Fig. 10 is about the same as that of Fig. 11. With the help of our progressive transmission and out-of-core algorithms, the scene images are rendered at 25 to 42 frames per second.

6. Conclusions

This paper presents an approach for transmitting and rendering large geographical maps on the network efficiently. Using graphics hardware and Voronoi diagrams of the geographical datasets, we achieve map simplification efficiently and preserve topological relations. Based on the simplification, we establish LOD map models. We also introduce a client/server architecture which integrates our out-of-core rendering algorithm and progressive transmission based on computer graphics hardware. Finally, we implement dynamically progressive transmission and reconstruction of large maps.

In this paper, the main deficiency of our method is that the encoding method used for transmission on the network can generate data redundancy. In the future, we will pay attention to this topic and research how to improve multiresolution controls and adaptive transmission for large terrain models and any complex objects residing on them.

Acknowledgements

This research was supported by National Natural Science Foundation of China (No. 60736007 and 60972128), NCET (No. NCET-07-0099), 973 Program (No. 2007CB714403), and the Open Project Program of the State Key Lab of CAD&CG (No. A0901), Zhejiang University, China. We also really thank the Editor and the reviewers for their kind suggestions.

References

- Asirvatham, A., Hoppe, H., 2005. Terrain rendering using GPU-based geometry clipmaps. In: Pharr, M., Fernando, R. (Eds.), *GPU Gems 2*. Addison-Wesley, pp. 27–44.
- Chandru, V., Rajan, V.T., Swaminathan, R., 1992. Monotone pieces of chains. *ORSA Journal on Computing* 4 (4), 439–446.
- Cignoni, P., Ganovelli, F., Gobbetti, E., Marton, F., Ponchio, F., Scopigno, R., 2003. planet-sized batched dynamic adaptive meshes, P-BDAM. In: *IEEE Visualization'2003 Proceedings*. pp. 147–155.
- Coors, V., 2003. 3D GIS in networking environments. *Computer, Environment and Urban Systems* 27 (4), 345–357.
- Danovaro, E., De Florian, L., Magillo, P., Puppo, E., Sobrero, D., 2006. Level-of-detail for data analysis and exploration: a historical overview and some new perspectives. *Computers & Graphics* 30 (3), 334–344.
- Douglas, D.H., Peucker, T.K., 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer* 10 (2), 112–122.
- El-Sana, J., Sokolovsky, N., 2003. View-dependent rendering for large polygonal models over networks. *International Journal of Image and Graphics* 3 (2), 265–290.
- Guilbert, E., Saux, E., 2008. Cartographic generalisation of lines based on a B-spline snake model. *International Journal of Geographical Information Science* 22 (8), 847–870.
- Guthe, M., Klein, R., 2004. Streaming HLODs: an out-of-core viewer for network visualization of huge polygon models. *Computers & Graphics* 28 (1), 43–50.
- Huang, B., Jiang, B., Li, H., 2001. An integration of GIS, virtual reality and the internet for visualization, analysis and exploration of spatial data. *International Journal of Geographical Information Science* 15 (5), 439–456.
- Kim, J., Lee, S., Kobbelt, L., 2004. View-dependent streaming of progressive meshes. In: 2004 International conference on shape modeling and applications (SMI 2004), 7–9 June, Genova, Italy. IEEE Computer Society, Silver Spring, MD, pp. 209–220.
- Kolar, J., 2006. On the road to 3D geographic systems: important aspects of global model mapping technology. In: *Innovations in 3D Geo Information Systems*. Springer Verlag, pp. 207–223.
- Lindstrom, P., Pascucci, V., 2002. Terrain simplification simplified: a general framework for view-dependent out-of-core visualization. *IEEE Transactions on Visualization and Computer Graphics* 8 (3), 239–254.
- Losasso, F., Hoppe, H., 2004. Geometry clipmaps: terrain rendering using nested regular grids. *ACM Transactions on Graphics* 23 (3), 769–776.
- McArthur, D.E., Fuentes, R., Devarajan, V., 2000. Generation of hierarchical multiresolution terrain databases using wavelet filtering. *Photogrammetric Engineering & Remote Sensing* 66 (3), 287–295.
- Mustafa, N., Krishnan, S., Varadhan, G., Venkatasubramanian, S., 2006. Dynamic simplification and visualization of large maps. *International Journal of Geographical Information Science* 20 (3), 273–320.
- Olsen, L., Samavati, F.F., Bartels, R.H., 2007. Multiresolution for curves and surfaces based on constraining wavelets. *Computers & Graphics* 31 (3), 449–462.
- Pajarola, R., Widmayer, P., 2001. Virtual geospatial exploration: concepts and design choices. *International Journal of Computational Geometry and Applications* 11 (1), 1–14.
- Rueda, A.J., Miras, J.R., Feito, F.R., 2008. GPU-based rendering of curved polygons using simplicial coverings. *Computers & Graphics* 32 (5), 581–588.
- Yang, B.S., Purves, R., Weibel, R., 2007. Efficient transmission of vector data over the internet. *International Journal of Geographical Information Science* 21 (2), 215–237.
- Yang, L., Zhang, L.Q., Kang, Z.Z., Xiao, Z.Q., Peng, J.H., Zhang, X.M., Liu, L., 2010. An efficient rendering method for large vector data on large terrain models. *Science China Information Sciences* 53 (6), 1122–1129.
- Zhang, L.Q., Yang, C.J., Tong, X.H., Rui, X.P., 2005. Visualization of large spatial data in networking environments. *Computers & Geosciences* 33 (9), 1130–1139.