International Journal of Applied Earth Observation and Geoinformation 12 (2010) 439–447

Contents lists available at ScienceDirect

International Journal of Applied Earth Observation and Geoinformation

journal homepage: www.elsevier.com/locate/jag

Web-based terrain and vector maps visualization for Wenchuan earthquake

Liqiang Zhang [a,*], Zhizhong Kang [b], Jonathan Li [c], Ling Yang [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] Department of Geomatics, School of Land Science and Technology, China University of Geosciences, Beijing 100083, China
[c] Department of Geography & Environmental Management, Faculty of Environment, University of Waterloo, 200 University Avenue West, Waterloo, Ontario N2L 3G1, Canada

ARTICLE INFO

Article history:
Received 3 August 2009
Accepted 12 January 2010

Keywords:
Wenchuan earthquake
Visualization
Progressive transmission

ABSTRACT

This paper presents a Web-based three-dimensional Geographic Information System (3DGIS) for Wenchuan earthquake disaster assessment. With the help of information technology resources, geoscientists are in a position to learn more about the structure of the earthquake in efficient ways. Due to huge spatial datasets of Wenchuan, China and narrow network bandwidth, general-purpose applications are difficult to transmit and visualize these datasets on the network. The application aims to interactively represent and transfer large spatial objects of Wenchuan County, China, as well as for dynamically rendering them in networking environments. Level-of-detail (LOD) terrain models and vector maps are created, and the server–client architecture is presented. The application provides an effective way for powerful access and manipulation of large-scale Wenchuan datasets.

1. Introduction

On May 12, 2008, Wenchuan of China happened a serious earthquake. It caused much great damage in 42 counties in Sichuan province, 23 counties in Gansu province, and 19 counties in Shanxi province. Other provinces such as Chongqing, Yunnan, Shanxi, Guizhou and Hubei provinces also have been influenced in this earthquake.

We have developed an earthquake application system which could provide functionalities for disaster information management, visualization and statistical analysis. It utilizes Web-based 3DGIS environments to manage the disaster database and visualize the monitored and analysis results. For earthquake disaster situation assessment, we need to integrate various remote sensed and aerial imagery and GIS data into an application for making decisions. The volume of these datasets is large, and is often distributed in remote computers. In order to visualize efficiently these earthquake-related spatial datasets and provide a good 3D user graphical interface for earthquake decision makers, transmission and visualization of DEM and vector maps are discussed in this paper.

2. Related work

2.1. Multiple representations of 3D geographical models

Lindstrom and Pascucci (2002) presented a method called SOAR for efficient out-of-core management and rendering of continuous adaptive terrain representation. The refinement framework is easy to display huge terrain datasets at high interactive frame rates. Cignoni et al. (2003) used a technique called P-BDAM for out-of-core management and interactive rendering of planet-sized terrain surfaces. It is said that it achieves better peak performance than SOAR on the same machine. Yang et al. (2005) combined TIN with Grid to construct multiresolution terrain models. They developed a method for encoding and storing vertex topological relationships in multiresolution terrain models. The connectivity among the triangulations needs to be stored and thus the method consumes lots of computation.

Estkowsk and Mitchell (2001) presented the complexity for maintaining topological relations and used the "detours" to eliminate the wrong intersections based on Douglas–Peuker polyline simplification method. Mustafa et al. (2001, 2006) proposed a Voronoi-diagram-based algorithm, which took advantage of the graphics hardware. It has a higher computational efficiency, and avoids the wrong intersections between the output lines, but this method cannot avoid self-intersections. Mantler and Snoeyink (2000) also presented an algorithm using the Voronoi diagram. They divided a complex polyline into a

* Corresponding author.
E-mail address: zhanglq@bnu.edu.cn (L. Zhang).

0303-2434/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.jag.2010.01.001

collection of safe sets. Each safe set could be simplified by using either a single line segment, or a standard polyline simplification algorithm, such as Douglas–Peuker polyline method, without introducing the intersections or topological changes. Zhang et al. (2005, 2009) proposed level-of-detail techniques for transmitting and visualizing spatial datasets. The techniques can real-time render large GIS datasets or models, but the efficiency needs to be improved further.

2.2. Methods for large 3D spatial data transmission

Several techniques have been proposed to compress and stream 3D terrain geometry. Bengt-Olaf and Ioana (1999) proposed a network graphics framework in which various transmission methods for 3D data was integrated into a single system. The framework did not take into account view-dependent refinement. Huang et al. (2001) presented a framework for interactive Web-based visualization by using 3D-based hybrid Internet computation through the simulation steering technique. However, the framework has difficulty to handle huge amounts of datasets. A client–server-based approach with a central scene description has been presented by Teler and Lischinski (2001). In order to response the clients' requests as soon as possible, the application transfers an impostor-based representation of the objects of the scene. Those 3D environments depend heavily on network connections, so it is an important design criterion to represent scene information in an efficient way to keep network traffic as low as possible. In order to adapt the immense data effort to the capabilities of the devices and the networks, Sahm et al. (2004) implemented the scene representation including a progressive data format with the help of an element graph and a progressive simplification algorithm. This algorithm traverses all element graphs of the given 3D scene, and separates the information of a scene element into several streams according to the element's data types. Coors (2003) introduced a query oriented data model for 3D geometry and topology multiple representations. Together with the P-tree spatial index, these multiple representations have been used to visualize 3DGIS data in a distributed networking environment. The Cyberwalk Web-based Distributed Virtual Environment (Chim et al., 2003) proposed a comprehensive solution that includes multiresolution caching of mesh geometry and a prefetching technique based on a prediction of the viewer's movements. Royan et al. (2003) presented an approach they developed. They introduced a method for dynamic network delivery of the complex models to user workstations during real-time sessions.

Buttenfield (2002) developed a progressive transmission method for vector maps. The simplification method first pre-computes several levels of the datasets on the server and put them into a modified strip tree for future delivery. The method is easy to deal with linear data sets at a given level-of-detail. However, this model does not support vertical links for efficient navigations between multiple representations, and the topological relations of the simplified ones are not explicitly preserved. On the other hand, due to the intrinsic complexity of the simplification and refinement operators, it is hard to implement the datasets transmission dynamically (Weibel and Dutton, 2005). Sester and Brenner (2004) described a set of generalization algorithms for visualization on small mobile devices. They used the algorithms to progressively transmit the building ground plans. The operations are effective, but they are not suitable to transmit other types of datasets. In order to transmit large vector maps on the Internet, Yang et al. (2007) exploited the constrained remove operations of vertices. Reconstruction was implemented through a process of vertex insertion on the client devices.

3. Definitions of specific terms

Our approach is mainly based on computer graphics methodologies. In order to present it clearly to readers who research in Geoscience fields, definitions of several computer graphics terms are given (http://en.wikipedia.org/wiki/).

Frame buffer: It is a video output device that drives a video display from a memory buffer containing a complete frame of data.
Depth buffer: A depth buffer is a buffer that is the same width and height as the render target. It records the depth of each pixel that is rendered.
Stencil buffer: A stencil buffer uses part of the depth buffer. It allows the programmer to set a stencil function that will test the "reference" stencil value against the value already in the stencil buffer each time a pixel is rendered.
View frustum: It is the region of space in the modeled world that may appear on the screen. The exact shape of this region varies depending on what kind of camera lens is being simulated, but typically it is a frustum of a rectangular pyramid.

4. Application architecture

We developed a Web-based 3DGIS application for Wenchuan earthquake. The application aims at visualizing the spatial datasets of Wenchuan region acquired from the server, evaluating the earthquake losses, and decision making for emergency response. Our application is composed of such sub-systems as database, analysis sub-system and decision-making sub-system for earthquake response. Fig. 1 illustrates the framework of our application.

The database stores such datasets related to potential Wenchuan earthquake hazards as GIS vector maps, DEM and imagery of Wenchuan region. The datasets are acquired by the client using the spatial index technique from the database, and then transmitted progressively to the client. The transmitted datasets are visualized in an integrated 3DGIS environment. The modules of earthquake hazard simulation, earthquake assessment and emergency response simulation can be manipulated on the 3D interface. The earthquake assessment module can be used to analyze the data and information, and calculate the results for further application in emergency response and/or post-quake recovering, such as earthquake damage assessment and route choice. The emergency response simulation provides users with tools for optimizing the decisions based on the results given by other sub-systems and the preset target. The Web-based 3DGIS system is aimed to provide users a 3D visualized platform for easy communication among the modules and a friendly interface.

4.1. Adaptive level-of-detail terrain models

In order to adapt DEM large data effort to the capabilities of the clients, the M-band wavelet transforms are applied to organize hierarchical terrain models.

4.1.1. The modified quadtree for level-of-detail terrain models

We have developed an algorithm to visualize the scene from the viewpoints, which can generate multiresolution terrain surfaces within a given geometric error, and in a manner that can preserve the critical features of the terrain.

Wavelet transforms have provided a more rigorous mathematical framework. In our approach, different M-band wavelet transforms are applied to remove vertices from high-resolution terrain datasets and to generate a set of from coarse to fine triangles depending on the distance to the viewpoint and the roughness of the terrain. We give an M-band wavelet family that

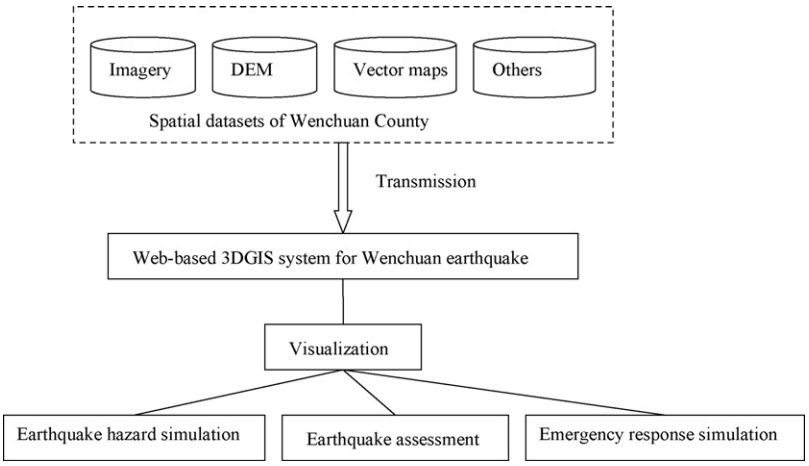


Fig. 1. Flowchart of intelligent simulation system for earthquake disaster assessment.

suits to represent DEMs and imagery. The low frequency component $\{a_{j,k,l}\}$ of the $j+1$ level M-band wavelet decomposition of image $\{a_{0,k,l}\}$ is depicted (Wan and Zhu, 1999; Zhang et al., 2005) as follows:

$$a_{j,k,l} = \sum_{n1}\sum_{n2} c_{n_1-Mk} c_{n_2-Ml} a_{j+1,n1,n2} \tag{1}$$

where j, k and l are integers; $n_1 = 0, 1, 2, \ldots, rows$, $n_2 = 0, 1, 2, \ldots, cols$; $rows$ is the number of the rows of DEM, and $cols$ the number of the columns of DEM.

The high frequency component $\{b^{t,s}_{j,k,l}\}$ of j level:

$$b^{s_1,s_2}_{j,k,l} = \begin{cases} \sum_{n1}\sum_{n2} c_{n_1-Mk} d^{s_2}_{n_2-Ml} a_{j+1,n1,n2} & s_1 = 0,\ 0 < s_2 < M \\ \sum_{n1}\sum_{n2} d^{s_2}_{n1-Mk} c_{n_2-Mk} a_{j+1,n_1,n_2} & 0 < s_1 < M,\ s_2 = 0 \\ \sum_{n1}\sum_{n2} d^{s_1}_{n1-Mk} d^{s_2}_{n_2-Mk} a_{j+1,n_1,n_2} & 0 < s_1,\ s_2 < M \end{cases} \tag{2}$$

The low frequency $\{a_{j+1,k,l}\}$ of $j+1$ level terrain datasets can maintain the important features of j level data $\{a_{j,k,l}\}$. So $\{a_{j+1,k,l}\}$ is an approximation of the original data.

In Fig. 2, the terrain surface in the view frustum is divided into three sub-regions $A_1B_1C_1D_1$, $A_2B_2C_2D_2$ and $A_3B_3C_3D_3$ based on the distance to the viewpoint. The width and length of the sub-region are 1/3 of the original. Since the region $A_1B_1C_1D_1$ is the closest to the viewpoint. The 2-band wavelet transform is applied to the original data set of the region to obtain an approximation. The region $A_2B_2C_2D_2$ which is a little far away from the viewpoint, so the 3-band wavelet transform is used to approximate the original data set of the region. $A_3B_3C_3D_3$ is the farthest away from the viewpoint, and approximation is generated by the 4-band wavelet transform. When the viewpoint is near to the terrain surface, the

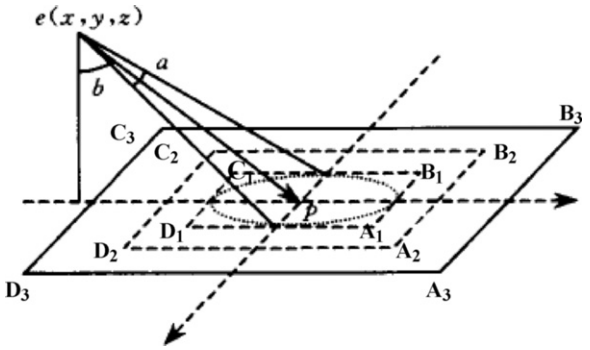


Fig. 2. Terrain surface tessellation is in the view frustum.

proper M-band inverse wavelet transforms reconstruct the hierarchical terrain models for creating a set of higher level of approximations. The same approach of the M-band wavelet transforms is also applicable to the image process.

In the subsequent steps, the hierarchical triangulation technique uses the filtered datasets to build multiresolution terrain surface. The whole terrain scene is built by a root triangulation, and then the triangulation is continuously divided until the height error is smaller than the given height error.

We should ensure that the projection d to the x–y plane of the triangulation mesh is smaller in size than one pixel h. If $d < h$, the terrain in this region is not rendered. If the neighboring tiles are simplified by the same band wavelet transforms, the tiles do not create gaps, otherwise, at the boundary of the tiles, cracks and shading discontinuities are emerged. Gaps can be avoided by ensuring that the projection to the x–y plane of an edge of one triangle contains no vertices of the projection of other triangles.

4.2. Vector map simplification

4.2.1. A simplification method based on Voronoi diagram

Douglas and Peucker (1973) proposed a polyline simplification algorithm. Their algorithm can maintain important geometric characteristics of polylines. However, topological relationships may change, resulting in three topological inconsistencies as shown in Fig. 3. In their method, if the polyline c' is the simplification result of the polyline c, the distance between c' and c is shorter than a given tolerance ε. All the shortcut segments of c can be divided into three groups: the shortcut segments belonging to the first group are approximate to c. Approximation is defined as the distance between $\overline{v_i v_j}$ and the original line segments $(\overline{v_i v_{i+1}}, \ldots, \overline{v_{j-1} v_j})$ is shorter than ε. So these shortcut segments also lie inside the ε-Voronoi diagram of c. In Fig. 4(a), this kind of shortcut segments is represented by the dashed lines. The shortcut segments in the second group (the dotted lines in Fig. 4(a)) also fall into the ε-Voronoi diagram of c, but are not approximate to c. The ones in the third group are outside of the ε-Voronoi diagram. The shortcut segments in the first and second groups are regarded as the compliant shortcut segments (the dashed lines in Fig. 4(b)) and the third groups are non-compliant shortcut segments (the dotted lines in Fig. 4(b)). All the approximate shortcut segments of a vector map are defined as S, and the approximate shortcut segments of c_i as s_i. The set s_i initially includes all shortcut segments of c_i.

All the non-compliant shortcut segments can be removed by the frame buffers and Voronoi diagram, resulting in the remaining shortcut segments far less than the initial ones. Then we can

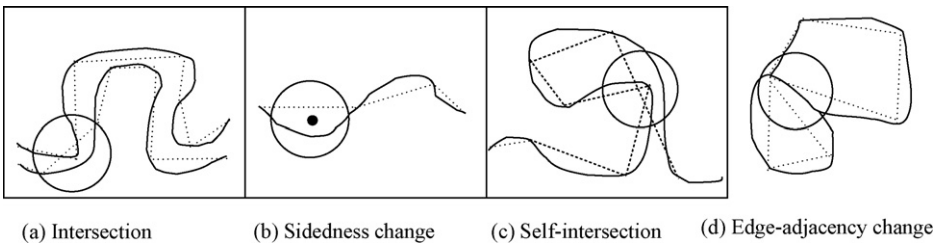


Fig. 3. Incompatible topological relationships. The solid lines are the initial features. The dashed lines are the simplified. The bold circularities show the place where the topological errors occurred. (a) Intersection, (b) sidedness change, (c) self-intersection and (d) edge-adjacency change. (a), (b) and (c) are taken from Mantle's safe set.

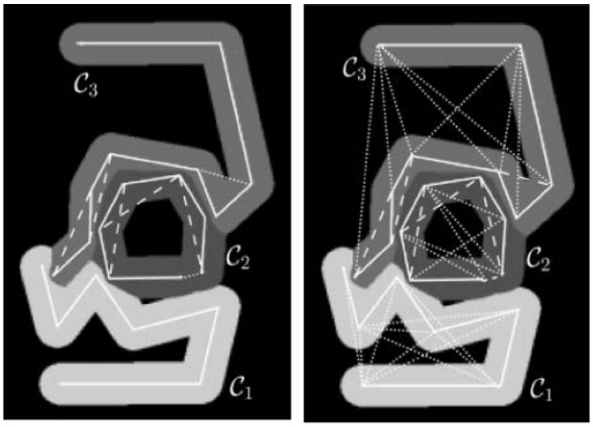
determine the distance between each remaining shortcut segment and c. Once the distance is larger than ε, we remove this shortcut segment. The remaining ones are thus all in the first group. Finally, connect them and get the simplified polyline c' by Douglas–Peuker algorithm (1973). The procedures for removing non-compliant shortcut segments are as follows:

(1) Empty all frame buffers.
(2) Render the ε-Voronoi diagram (Hoff et al., 1999) of the whole vector map in the stencil buffer.

    In this step, each polyline's ε-Voronoi region has a unique value in the stencil buffer. For the polyline c_i, i.e., its ε-Voronoi region's stencil buffer value is i. Then draw all the shortcut segments with a unique color. For each polyline c_i, set the stencil test to pass only if the stencil value is equal to i.
(3) Repeat the process for all the polylines. If a shortcut segment of c_i is beyond its polyline's ε-Voronoi region of c_i, it appears in the color buffer. Once a shortcut segment's color is found in the color buffer, the shortcut segment is non-compliant. In this way, we can remove all the non-compliant shortcut segments.

4.2.2. Occluding problem

If the whole vector map is rendered in the frame buffers, each vector feature is significantly zoomed out, even just occupies several pixels. In this case, a segment may be occluded by other segments so that we cannot identify or remove all non-compliant shortcut segments at a time. Usually, the short segment formed by two vertices far away from each other is likely to exceed the ε-Voronoi zone. In that condition, we can just link a vertex with the nearer m (m is a positive integer) vertices to create shortcut segments. If the sample interval of the map is small, the variable m should be set larger. Otherwise, m can be smaller.



(a) Set of compliant proximate shortcut segments (dashed) and compliant non-proximate segments (dotted)

(b). Set of compliant shortcut segments (dashed) and non-compliant shortcut segments (dotted).

Fig. 4. Different kinds of shortcut segments (Mustafa et al. (2006). (a) Set of compliant proximate shortcut segments (dashed) and compliant non-proximate segments (dotted). (b) Set of compliant shortcut segments (dashed) and non-compliant shortcut segments (dotted).

Compared with the geometry algorithms, our method preserves topological consistency efficiently, and avoids dealing with unnecessary shortcut segments. In the following section, we present how to keep the topological relationships.

4.2.3. Preservation of topological consistency

The first topological inconsistency is wrong intersections among simplified features (see Fig. 3(a)). As shown in Fig. 5(a), the distance between the shortcut segment $\overline{v_i v_{i+2}}$ and c_1 is shorter than ε. If we just use Douglas–Peuker polyline simplification algorithm (1973), $\overline{v_i v_{i+2}}$ may be contained in the simplification result. Thus, the simplified result of c_1 intersects with the polyline c_2 wrongly. By rendering the Voronoi diagram, $\overline{v_i v_{i+2}}$ passes the ε-Voronoi region of c_2. In other words, it is beyond the ε-Voronoi region of c_1. Therefore, it is removed from S. In this way, the wrong intersections are avoided.

The second topological consistency is called point constrains (see Fig. 5(b)), which means that if point features are involved in a vector map, each point should locate at the same edge of the polyline c_i and its simplified result c_i'. This problem can be solved by the stencil buffer as well. After rendering the ε-Voronoi diagram of all the polylines, render the ε-buffers of all point features in the stencil buffer. During this process, the depth test is closed in order to make the points' buffers occlude the original Voronoi diagram. All shortcut segments which pass the ε-buffers of the point features are removed from S. In Fig. 6, the shortcut segment $\overline{v_i v_{i+2}}$ passes the ε-buffer of the point p_i, so it should be removed from S.

Self-intersections are shown in Fig. 3(c). The method proposed by Mustafa et al. (2006) cannot avoid this topological inconsistency. We solve this problem by using the stencil buffer. A polyline's self-overlapped ε-Voronoi region may cause self-intersections (see Fig. 6). So we remove the self-overlapped ε-Voronoi region to avoid self-intersections. A strip of quadrangles is applied to approximate a polyline's ε-buffer. In the same time, quadrangle strip's overlapped regions are used to approximate the self-overlapped ε-Voronoi region (see Fig. 7).

After rendering the ε-Voronoi diagram, draw the quadrangle strips for all polylines in the stencil buffer as well. For the polyline c_i, the stencil value of its ε-Voronoi region is i. When drawing the quadrangle strip, we set the stencil test to pass only if the stencil value is no less than i. And configure the stencil operation to increase the stencil value when a quadrangle is rasterized. After rendering, if the stencil value is larger than $i+1$, the ε-Voronoi region is overlapped on this pixel. Otherwise, this pixel's stencil value is equal to $i+1$. Finally, draw the shortcut segments, and set the stencil test to pass only if the stencil value is equal to $i+1$.

In addition, there is another topological relationship need to be preserved, which is called the adjacent relationship as shown in Fig. 3(d). If two adjacent polygons share an edge and these polygons are simplified respectively, the edge will be simplified twice, which leads to create two different lines. Therefore, the

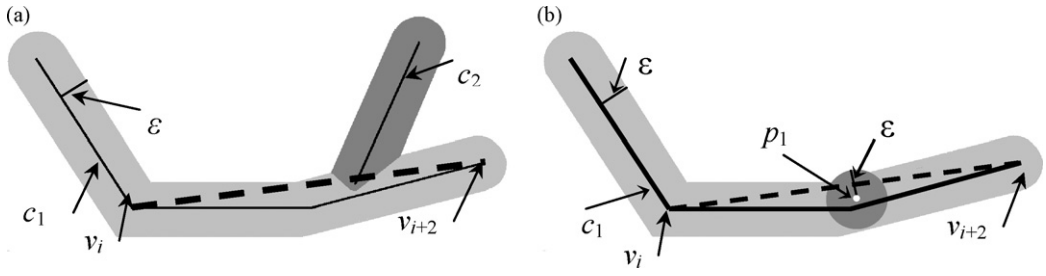


Fig. 5. (a) Using ε-Voronoi diagram to avoid intersections between polylines. The region colored light gray is ε-Voronoi region of the polyline c_1. And the dark gray region is ε-Voronoi region of the polyline c_2. (b) Using points' ε-buffers to keep points' sidedness. The dark gray round is the ε-buffer of the point P_1.

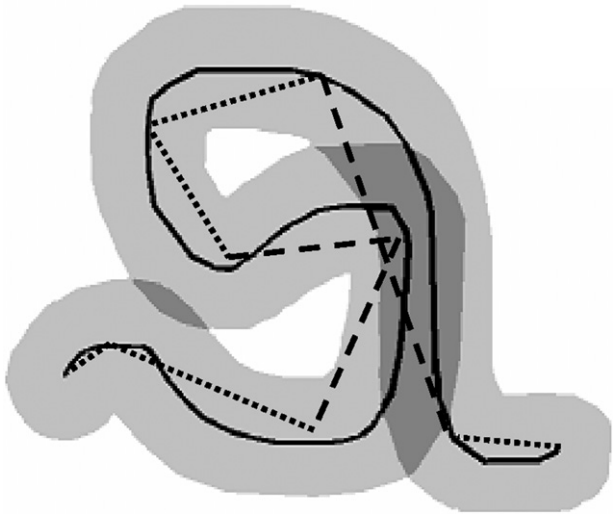


Fig. 6. Self-overlapped ε-Voronoi regions (dark gray). The shortcut segments in these regions (dashed) may cause self-intersections. Others (dotted) will not.

boundaries of these two polygons are no longer superposed. To make for this kind of mistake, the edges must be simplified once. In other words, we should simplify the edges which compose the polygons' boundaries, instead of simplifying the boundary as a whole.

4.3. Transmission and reconstruction

Generally, implementation of our application is divided into two procedures: one is the terrain and 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.

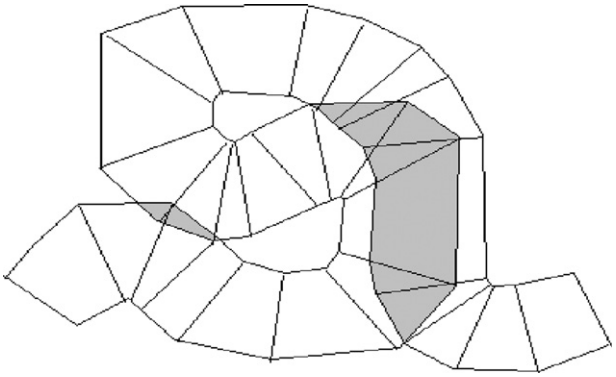


Fig. 7. The quadrangle strip used to approximate the ε-buffer of a polyline. And the overlapped regions (gray).

During navigation and interaction on the client, the rendering operation firstly acquires the subsets in a prioritized order from the server and loads them onto the graphics card. For the first procedure, most of the time is spent waiting for the server's response, which comprises the network latency and the time for multiresolution hierarchy traversal on the server.

Fig. 8 illustrates the architecture of our application. The architecture is composed of three components: the client, the application server and the database/files. The client is responsible for reconstructing the progressive data transferred from the server and rendering them dynamically. The database/files store the multiresolution models of the spatial datasets, and the server traverses the hierarchy to select map updates for the client. 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 simplification process of the server simplifies the retrieved maps and 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 users, the process stops.

4.3.1. A prefetching mechanism

The basic principle of the prefetching is to predict which models will be downloaded to the client, or the models on the local hard disk will be cached in to the graphical card with some of look ahead. Prediction is required to determine the next model that the application will need. Generally, a user navigating in the scene often causes the view position change, the navigation direction is not known in advance accurately. When browsing across a map boundary, or zooming in past a critical resolution threshold, will trigger the server's response for retrieving next frame models. In order to retrieve these datasets in advance, the server need get the information, such as the positions of the eye and the target on the landscape and the view direction from the client. When the next frame is to be rendered, the client has predicted the view parameters for the next frame and sent the parameters to the server. Therefore, the next frame datasets will have already been delivered to the client in advance and are ready to be applied.

The eye point E_n, target point R_n and view direction L_n of the next frame can be predicted based on the parameters of the previous and current frame (see Eq. (3)).

$$\begin{aligned} E_n &= E_c + \mu(E_c \vec{-} E_p) \\ R_n &= R_c + \rho(R_c \vec{-} R_p) \\ L_n &= R_n - E_n \end{aligned} \tag{3}$$

where E_c and E_p are respectively the eye points of the current frame and previous frame, R_c and R_p are respectively the target points of

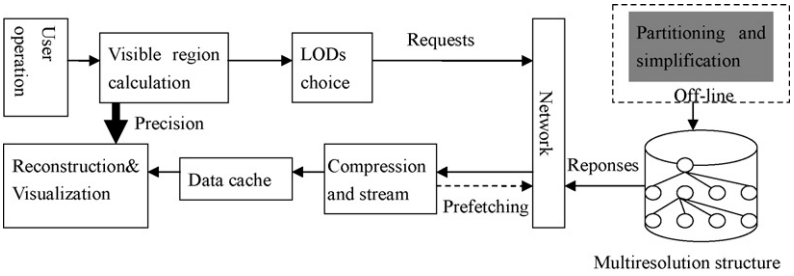


Fig. 8. Client/server architecture of the application.

the current frame and previous frame. ρ and μ are variables. Generally, $\rho = \mu = 1$.

The client initiates the prediction after the current view is loaded. By utilizing the prediction-and-prefetching, the client acquires the models from the server according to the current and predicted view positions, and then caches them into local memory. Computing the results of Eq. (3) is fast and occupies little memory. For regular user navigation, the predictive error usually deviates a little from the actual position. Therefore, the server can retrieve the correct tiles and models based on the predictive view parameters. When the navigation is fast, the inaccurate view-dependent operation caused by the prediction will not effect the fidelity of the scene since the human vision system cannot view much details of the scene in fast motions (Luebke et al., 2002).

4.3.2. Progressive transmission

The server is responsible for generating multiresolution models from the database, creating level-of-detail (LOD) map models off-line and generating 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 presents 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 client later uses the metadata file to determine the location of the vector maps. The metadata file and multiresolution models are placed on the location accessible to the Web server for later transmission to the client.

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 steams 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.

4.3.3. Reconstruction on the client

The XML metadata file is first transmitted to the client through a file transfer protocol. The client reads the metadata file and knows the current positions of the datasets from the file. If the models are on the client, the client sends a request for multi-resolution models, and the renderer operator on the client read the data from the cache or the hard disks. Otherwise, the client sends a request to the server for the multiresolution models. The request includes the current view parameters and the ones of the next frame. The client can pre-compute the view parameters for the next frame which is introduced in Section 4.3.1, and sends the predicted view parameters to the server as the map update for requesting the vertices that have not been stored by specifying the required LODs.

The client initially receives the coarser datasets from the server. By monitoring the process, the client can identify the portion of the scene 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 based on the viewpoint. It incorporates them into the data array as they arrive, progressively improves the accuracy of the maps being visualized. The downloaded data 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 information allows the system to prioritize parts of the scene that have the biggest visual impact. The prediction-and-prefetching strategy tries to keep as many vertices as possible in main memory. When the size of the queue reaches its capacity, the least recently used subsets are discarded from the priority queue to make room for new incoming datasets. Lower priority vertices that have stayed in memory long are removed from memory first, so that only data recently rendered or still within the extended view frustum remains in memory.

Some view change operations do not always require a new request. In cases where new datasets are not required, the client implements the change in view via model coordinate transformations. Otherwise, the rendering operation will capture the data from the server or from the local hard disks. On the other hand, network latencies are not acceptable. In order to compensate for these deficiencies, the prediction-and-prefetching mechanisms are used during data access on the network or from the hard disk. Prefetching reduces the latency during data access the network or disk. The client supports a multithreaded model so that multiple tiles may be requested at once. Each thread opens a connection back to the server for requesting the maps.

To speed up scene visualization on the client side, we develop multi-thread techniques to implement transmission and visualization based on the viewpoints and the view direction. Several threads including data prediction, refinement and rendering run in the back end of the application communicating with the external processes. The prediction thread mainly computes the next frame the client will render. This thread will be triggered when the client is idle. The refinement operation thread is invoked to compute the visible region once the viewpoint changes, responsible for vertex array coordinates update and sends the streaming to the client progressively. The rendering thread implements the multiresolution rendering and visualization. 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 landscapes.

5. Experimental results

In this section, we test the performance of our methods. The experiments are done on a personal computer with 3.00 GHz Intel (R) Pentium (R) 4 CPU, 768 MB main memory and Intel (R) 82865G Graphics controller graphic card. Our application system is built by using VC++.NET and OpenGL. In this application, we use Winsock to implement network transmission and visualize vector maps on the clients.

The datasets located in Sichuan province are consisted of 1:10,000 transportation map, 1:50,000 base geographical data, 1:50,000 DEM, Quickbird imagery of the prior earthquake, and aerial imagery with 0.5 m resolution. Other datasets includes: (1) urban and rural residential areas; (2) urban and rural infrastructures such as industrial and mining lands, highways, railways, bridges, electric facilities, dikes and reservoirs; (3) geological and environmental change information such as earthquake lakes, rock slumps, landslides and turbidities; and (4) destroyed farmlands such as croplands and forest lands.

Table 1
Three test vector datasets.

Dataset	Type	No. of vertices	Data volume (kb)
1	Transportation map	25,220	394
2	Urban and rural residential polygons	90,145	1408
3	Stream and lake map	987,204	15,425

5.1. Progressive transmission

The experiments are undertaken in an LAN with a network bandwidth 2 Mbps. Three vector maps are used (see Table 1). We have computed the simplification time, transmission time, render time and reconstruction time under different compression ratio of the three maps. The results are illustrated in Table 2.

For Dataset 1, our proposed approach cannot make the transmission procedure faster than the method without progressive transmission (see Table 2). This reason is that the data volume

Table 2
Comparison of the waiting times and reconstruction times with/without progressive transmission.

Vertices number	Data volume (kb)	Compression ratio (%)	Simplification time (s)	Transformation time (s)	Waiting time (s)	Reconstruction time (s)	Render time (s)
1							
Progressive transmission							
1033	16	4.1	4.1	0.2	4.4	3.9	0.02
4562	71	18.1	1.7	0.8	2.6	3.6	0.08
12,826	200	50.8	1.2	2.3	3.7	2.0	0.16
Without progressive transmission							
25,220	394	0	0	4.0	4.2	0	0.23
2							
Progressive transmission							
8903	139	9.9	5.2	1.9	7.7	15.3	0.60
34,387	537	38.1	1.6	5.6	8.1	11.1	0.88
Without progressive transmission							
90,145	1408	0	0	14.6	15.8	0	1.22
3							
Progressive transmission							
109,434	1709	11.1	15.1	22.9	40.7	230	2.72
411,251	6425	41.7	6.2	66.4	78.4	178	5.83
Without progressive transmission							
987,204	15,425	0	0	156.9	170.9	0	13.89

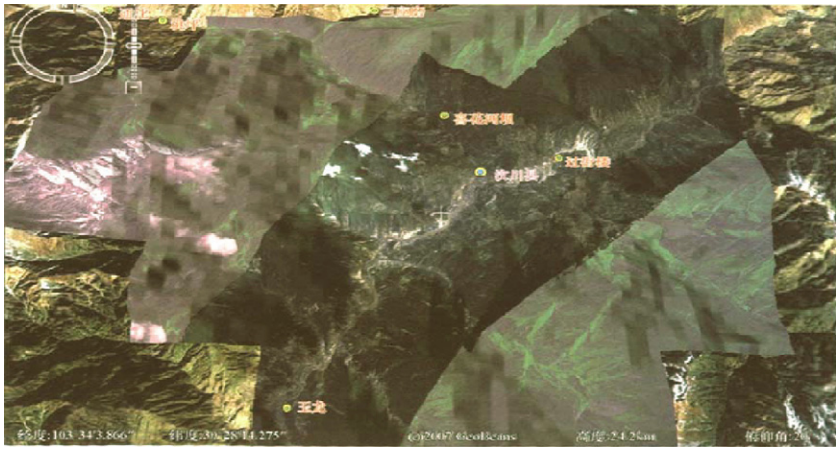


Fig. 9. Rapid imaging data released and multiresolution images superposition displayed.

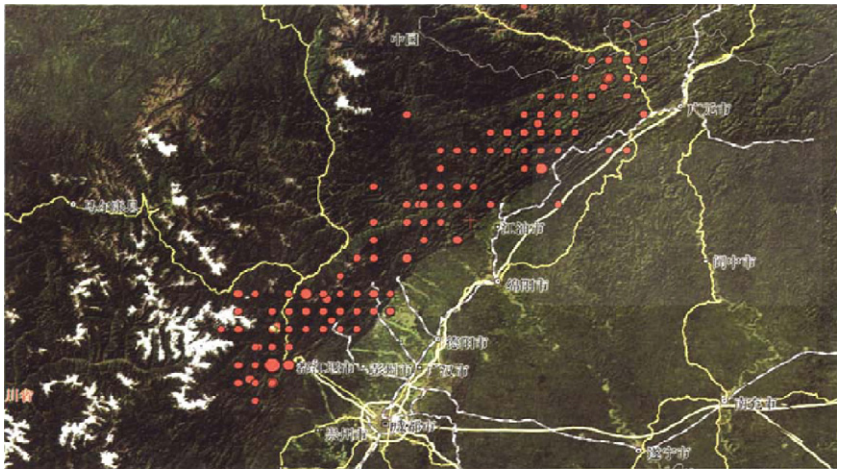


Fig. 10. Distribution of earthquake in Wenchuan.

is small, and data simplification and reconstruction cost some time. Therefore, these processes make the rendering speed slower than the latter method, when the network bandwidth is 2 Mbps. With data size increasing (such as Dataset 3), the simplification and reconstruction process need much less time compared with the transferring process, so our method can improve the efficiency of data transmission and visualization. Moreover, our method reduces the network latency. From Table 2, we can also see that the simplification time and transmission time can be well balanced, the transmission on the network will be faster and the server-side burdens are not large. Since polygons of data models and their topological relationships are preserved, users querying the datasets, for example for statistical data describing neighboring districts, can get the same results.

5.2. Overlaying vector maps on 3D terrains

In our application, the terrain is the base model that will carry the rest of the man-made environment, such as cities, roads, bridges, or additional georeference information. Vector maps are usually rendered on the terrain surfaces. Terrains are generated based on hierarchical LOD models. Therefore, the view-dependent LOD vector maps are overlapped on the multiresolution terrains surfaces. Based on the two terrain models, we overlap the vector maps on the terrain surfaces respectively (see Figs. 9 and 10).

From what has been mentioned above, we can come to the conclusion that our method can simplify and overlap vector maps on the LOD terrain models efficiently, while keeping topological consistency of vector features.

6. Conclusion

A Web-based 3DGIS application for earthquake disaster assessment can help authorities responsible for earthquake disaster management as well as post-earthquake recovering. The application presented in this paper has been used to identify earthquake damage assessment and post-earthquake emergency response in Wenchuan.

In this paper, we emphasis on an efficient server/client architecture for representing and transmitting spatial data on the network. Based on the multiresolution hierarchical models for terrain surface and vector maps, our framework integrates the progressive transmission techniques to reduce the perceived network delay, and then applies multiple representations to visualize the massive spatial datasets on the clients. All these schemes make the application interactive operation of the terrain and multiple collections of the buildings in the networking environments possible.

In the future work, we consider extending the proposed techniques to represent and transmit more complex 3D models, such as geological data and irregular architectural buildings in networking environments.

Acknowledgements

This research was funded by NationalNatural Science Foundation of China (No. 60736007 and 60972128).

References

Bengt-Olaf, S., Ioana, M., 1999. An adaptive framework for 3D graphics over networks. Computers & Graphics 23 (6), 867–874.

Buttenfield, P., 2002. Transmitting vector geospatial data across the Internet. In: Egenhofer, M.J., Mark, D.M. (Eds.), Lecture Notes in Computer Science, vol. 27, no. 48, pp. 51–64.

Coors, V., 2003. 3D GIS in networking environments. Computer, Environment and Urban Systems 27 (4), 345–357.

Chim, I.J., et al., 2003. A Web-based distributed virtual walkthrough environment. IEEE Transactions on Multimedia 5 (4), 503–515.

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.

Douglas, H., Peucker, K., 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. The Canadian Cartographer 10, 112–122.

Estkowsk, R., Mitchell, J.S.B., 2001. Simplifying a polygonal subdivision while keeping it simple. In: 17th ACM Symposium on Computational Geometry. pp. 40–49.

Hoff, K. E., Culver, T., Keyser, J., Lin, M. and Manocha, D., 1999. Fast computation of generalized Voronoi diagrams using graphics hardware. Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques. New York: ACM Press/Addison-Wesley Publishing Co., 277–286.

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 (1), 439–456.

Luebke, D., Reddy, M., Cohen, J., Varshney, A., Watson, B., Heubner, R., 2002. Level of Detail for 3D Graphics. Morgan Kaufmann.

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.

Mantler, A., Snoeyink, J., 2000. Safe sets for line simplification. In: 10th Annual Workshop on Computational Geometry.

Mustafa, N., Koutsofios, E., Krishnan, S., Venkatasubramanian, S., 2001. Hardware-assisted view-dependent map simplification. In: Proceedings of the Seventeenth Annual Symposium on Computational Geometry. pp. 50–59.

Mustafa, N., Krishnan, N.S., Varadhan, G., Venkatasubramanian, S., 2006. Dynamic simplification and visualization of large maps. International Journal of Geographical Information Science 20, 273–320.

Royan, J., Bouville, C., Gioia, P., 2003. PBTree—a new progressive and hierarchical representation for network-based navigation in urban environments. In: Proc. Vision, Modeling, and Visualization. pp. 299–307.

Sahm, J., Soetebier, I., Birthelmer, H., 2004. Efficient representation and streaming of 3D scenes. Computers & Graphics 28, 15–24.

Sester, M., Brenner, C., 2004. Continuous generalization for visualization on small mobile devices. In: Fisher, P. (Ed.), Developments in Spatial Data Handling—11th International Symposium on Spatial Data Handling. Springer, Berlin, pp. 469–480.

Teler, E., Lischinski, D., 2001. Streaming of complex 3D scenes for remote walkthroughs. Eurographics 20 (3) (Manchester, UK).

Wan, G., Zhu, C.Q., 1999. Application of multi-band wavelet on simplifying DEM with lose of feature information. Acta Geodaetic Cartograph, Sinia 28, 36–40.

Weibel, R., Dutton, G., 2005. Generalizing spatial data and dealing with multiple representations. In: Longley, P., Goodchild, M.F., Maguire, D.J., Rhind, D.W. (Eds.), Geographical Information Systems: Principles, Techniques, Management and Applications. 2nd edition (abridged edition). Wiley, Hoboken, NJ, pp. 125–155.

Yang, 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, B.S., Shi, W.Z., Li, Q., 2005. An integrated TIN and Grid method for constructing multi-resolution digital terrain models. International Journal of Geographical Information Science 9 (10), 1019–1038.

Zhang, L.Q., Yang, C.J., et al., 2005. A web-mapping system for real-time visualization of the global terrain. Computers & Geosciences 31, 343–352.

Zhang, L.Q., Guo, Z.F., Kang, Z.Z., et al., 2009. Web-based visualization of spatial objects in 3DGIS. Science in China Series: Information Sciences 52 (9), 1588–1597.