International Journal of Digital Earth, 2014
Vol. 7, No. 1, 53–67, http://dx.doi.org/10.1080/17538947.2012.667159

Taylor & Francis
Taylor & Francis Group

Web-based visualization of large 3D urban building models

Liqiang Zhang[a]*, Chunming Han[b], Liang Zhang[a], Xiaokun Zhang[c] and Jonathan Li[d]

[a]*State Key Laboratory of Remote Sensing Science, Beijing Normal University, Beijing, China;* [b]*The Center for Earth Observation and Digital Earth, Chinese Academy of Science, Beijing, China;* [c]*Department of Computer Science, Beijing Electronic Science and Technology Institute, Beijing, China;* [d]*Department of Geography and Environmental Management, Faculty of Environment, University of Waterloo, Waterloo, Canada*

(*Received 27 June 2011; final version received 13 February 2012*)

Adaptive rendering of large urban building models has become an important research issue in three-dimensional (3D) geographic information system (GIS) applications. This study explores a way for rendering web-based 3D urban building models. A client–server hybrid rendering approach is presented for large 3D city models, stored on a remote server through a network. The approach combines an efficient multi-hierarchical building representation with a novel image-based method, 3D image impostor generated on demand by a remote server. This approach allows transferring complex scenes progressively while keeping high visualization quality. We also evaluated the rendering and data transferring performance of the proposed approach.

Keywords: urban building models; progressive transmission; client–server hybrid; 3D visualization

1. Introduction

Three-dimensional (3D) urban visualization has become a fast growing area of research for many scientists. Because of the complexity of urban models, which combines large extents with rich small-scale visual details, real-time visualization of virtual cities on the Internet is a challenging problem. Current rendering methods in the network environment can be classified into two types based on where the geometry-rendering task is implemented: server-based techniques and client-based ones. For the server-based architecture, once the number of connections exceeds a threshold, the workload of the server becomes very large and thus expensive. In the client-based architecture, the client usually performs the modeling and visualization tasks. When large city models are rendered on the client, the rendering process is often not interactive and visual quality is poor because of the limited performance of the client.

To adapt the geographical data to computing infrastructure including random access memory (RAM) size, central processing unit (CPU) power, graphics support, and bandwidth, we herein propose a client–server hybrid rendering approach. In this approach, the client renders the geometry of the city buildings that are

*Corresponding author. Email: zihaozhang2003@yahoo.com.cn

close to the viewer, and the distant buildings are rendered as impostors that will be rendered by the server and streamed to the client. Since impostors represent buildings distant from the viewer, they do not need to be updated unless the user's position in the virtual environment changes beyond a given threshold. The approach is also supported by a set of tools that dynamically split rendering tasks between the server and the client and take into account the computing capabilities of the client and the network load. This scheme allows viewers to view different regions interactively at different levels of detail (LODs) on the network and thus provides us with a great degree of control over the transmission.

2. Related work

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

2.1. Simplification of 3D building models

Generalization in 3D spatial objects is still considered in its infancy stage (Glander and Döllner 2009). An overview of 3D generalization issues can be found in Meng and Forberg (2007). A number of algorithms have been proposed (Li 2007) that employed different strategies to eliminate minor parts of a building and small features. More recent techniques simplify buildings in 3D, and these techniques are designed to find and eliminate small volumetric features such as protrusions on the surface model (Thiemann 2002, Forberg 2007).

Thiemann and Sester (2004) divided the whole generalization process into a segmentation, interpretation, and generalization procedure. The boundary representation of a building was segmented into a set of convex parts that were stored as a cell complex and a CSG-tree. The CSG-tree presents a hierarchical subdivision useful for generalization; however, automatic determination of semantic information during the segmentation is not well addressed. Anders (2005) proposed an algorithm for simplifying 3D urban models by aggregating nearby buildings. He projected building models onto three orthogonal planes and obtained simplified models based on the projections, but this method finds difficulties when non-symmetric models have the self-occlusion problem during the projections. Forberg (2007) introduced a scale-space approach to generate LOD representations of 3D city models; it works by moving parallel facets toward each other until the facets meet and merge. Therefore, this approach is only employed to deal with orthogonal building structures, but is not well suited when treating roofs, walls, and other non-orthogonal structures.

Although 3D scene fits with people's spatial awareness better than 2D maps, it also brings more problems as the third dimension of spatial psychology is introduced. Great attentions have been paid on landmark, path, and district in urban legibility (Lynch 1960). Landmark buildings has many characteristics: singularity, prominence, distant/local, accessibility, content, and prototypicality (Sorrows and Hirtle 1999). In most cases, landmarks are separated from other ordinary buildings, taking into account their different generalization methods. Many researches (Raubal and Winter

2002, Winter *et al.* 2008, Ganitseva and Coors 2010) detected landmark buildings by assessing their saliency: integrating visual, semantic, and structural properties in relation to neighborhood. Glander and Döllner (2008) partitioned 3D city models into cells formed by the street network, water, railway, and coast lines. Large blocks extruded from each cell footprint are used to replace individual models inside, while global landmark objects (identified by relative height) are dynamically exaggerated during the interactive visualization. An improved method was presented later by Glander and Döllner (2009) to automatically generalize hierarchical 3D city building models by creating several representations of increasing levels of abstraction. It appears that their researches are concentrated more on local landmarks, and not aiming at keeping the real appearance and skyline of original models. Chang *et al.* (2008) used the single-link hierarchy method to generate building clusters considering road information. To better preserve the skyline, they define distant landmarks as buildings taller than their parent generalized blocks, which are drawn separately at the cost of overlapping.

2.2. Web-based 3D visualization applications

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 finds it hard to handle large amounts of data-sets. Pajarola and Widmayer (2001) described a virtual reality prototype system, where a client runs the virtual reality (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. Royan *et al.* (2003, 2006) applied a Delaunay triangulation-based merging algorithm and a binary-tree structure called PBTree, to represent dense urban areas over the network. His approach allows users to navigate freely in the city scene. However, the generated PBTree has many levels that prevent viewers from selecting proper building models efficiently. Coors (2003) proposed a data model for 3D geometry and topology in 3DGIS. In this method, progressive transmission was used to visualize 3DGIS data in the networking environments together with a spatial index method called P-tree. El-Sana and Sokolovsky (2003) developed a web-based 3D scene system. To improve the adaptive visualization and reduce transmission time, the client uses cache and a prediction mechanism to obtain the data-set from the server. They subdivide the view-dependent tree into blocks which allow selective refinement and maintain the server-side with a list of active nodes for each client connected. The server responds to the client requests for sending the update operations needed to satisfy the visual query. Kim *et al.* (2004) came up with a client/server framework for view-dependent streaming of the progressive meshes: the servers send the base mesh and the vertex hierarchy of 3D models to the client, and then the client creates further requests for selective refinement operations, and the server continues to send the remaining LOD models until the requests stop. However, the application cannot support interactive exploration of models that usually involves selective refinement. Zhou and Tan (2006) presented a client/server architecture integrating the Internet GIS with multitier web applications. The architecture can

render 3D city models, including digital building models, digital elevation models (DEMs), and large-scale urban ortho-image in the network environment. Danovaro *et al.* (2006) also designed a client/server system for allowing an application to deliver a mesh progressively, which overcomes some limitations of some previous work. The client can carry out simple computations and store more than the mesh generated for the visualization process. It builds a partial copy of the multi-tessellation (MT) model as it receives information from the server. Okamoto *et al.* (2011) proposed a view-dependent rendering system for large-scale 3D models that are located on a remote server. Their system uses both model- and image-based rendering methods for efficient load balance between a server and clients. The disadvantage is that it is difficult to handle spatial analyses for the 3D models on the client side, as the models are displayed as images.

Google Earth is a 3D web-based visualization application of the planet. It integrates satellite imagery, 3D modes, maps, and the power of Google search into the GIS platform. It can run on PC in the current network environment. Skyline can combine the aerial imagery with other 3D spatial features to create an interactive landscape on the desktop or in the networking environments. The large data can only be stored in files and not in databases, and the data cannot be modified once they are integrated by TerraBuilder. Chinese Academy of Sciences has built Digital Earth Prototype System (DEPS), which is a highly integrated system with data services, information fusion and extraction, analysis, and application (Guo *et al.* 2009). The DEPS has been successfully applied in regional research fields, such as disaster mitigation, digital archeology, and digital city (Guo *et al.* 2010). It has generated a widespread influence in the digital earth field.

3. The hybrid approach

There are two main types of client–server cooperation (pull model and push model) for demand-driven transmission. Our approach is a pull strategy. In order to keep network traffic as low as possible and adapt to the client computing capabilities, we propose a client–server hybrid rendering approach. Our approach can split the rendering task dynamically between the server and the client. The client renders the geometry of the city building models close to the viewer. The distant buildings are clustered and simplified, and then they are portrayed as impostors rendered by the server and streamed to the client. Since impostors represent buildings distant from the viewer, they do not need to be updated unless the user's position in the virtual environment changes beyond a given threshold.

The server undertakes the following main tasks: (1) store the city building models; (2) generalize LOD representations of 3D city models close to the user's position and send them to the client for displaying; (3) simplify collections of 3D buildings, which is far from the users' current position, through generalizing and clustering the footprints of neighboring buildings, and render and supply to the client of pre-computed 3D imposters.

The client performs the following main tasks: (1) render the near city building models according to the required LOD, (2) display the distance buildings as 3D imposters, and (3) request the server updates for both the 3D models and the images to be displayed as the user changes its position.

3.1. Model-based transmission and rendering for near buildings

The complexities of urban building models vary greatly from each other. Usually, we simplify 3D building models by the geometry information like vertices, façade identifiers and areas, normal vectors and the spatial affiliations among neighboring buildings. Some building surfaces are special with tower roofs, chimneys, small pyramids, or any other polyhedrons which show the building models in detail. If the upper and lower surfaces of these types of structures are parallel to the ground and their areas are less than a given threshold, these structures are removed. The criterion to determine the sharp roofs and decorations is the angle between the surface and the ground. The angle is usually from 0 to 75° (from the experimental data-sets, we find the angle in most of the buildings falls into this range).

Complex buildings may contain several facades and floors parallel to the ground, and some facades' shapes are L-type or trapezoidal. Some roofs are parallel to the ground, but there may be small rectangular structures in the top center. Simplification can be achieved by moving the parallel meshes sequentially. It can be divided into two steps. First, we need to rectify some irregular facades. For example, in buildings like platform ladder whose facades are nearly vertical to the ground, critical angle (like 65°) is used to rectify them. Take a triangle, the basic cell of facades in .3ds model, as an example: the slopes of the three edges of the triangle are calculated, and then we pick out the edge which has the biggest slope. p_1 and p_2 are its two vertices and we change the coordinate value x, y of the two points, $p_1.x = p_2.x = (p_1.x + p_2.x)/2$, $p_1.y = p_2.y = (p_1.y + p_2.y)/2$ and the value of z which represents building height is unchanged. Then the parallel facades or floors are traversed. When the dot product of the standard vectors of the facades is 1, they are parallel facades. We define the outward direction of the counterclockwise vertices of the mesh as the positive direction. Next, the coplanar meshes of each group are stored by a dynamic 2D array. To determine the coplanar meshes, subtract two vertices taken from each of the meshes and calculate the distance between the parallel meshes. The last step is to shift the walls or floors with the minimum distance in each building and update the array that stores the identifiers of the coplanar meshes and the distance array. We merge the two surface sets with the minimum distance according to the distance between the non-coplanar meshes, and decide the displacement mainly on the area of the buildings. When the distance between two groups of the parallel meshes is d, the area of the first group and second group is $area_1$ and $area_2$, respectively. Here, we discuss in three situations according to the relationships between $area_1$ and $area_2$. If $area_1 > area_2$, the displacement of the first group is equal to zero, and the displacement for the second group is equal to d, the meshes owing the large area are not moved; the process is similar when $area_1 < area_2$. When $area_1 = area_2$, take the normal vector as the criterion. Along the positive direction of the normal vector, keep the meshes in the front unchanged and move the rear meshes. Next, merge the two nearest parallel meshes, thus complex building models are simplified progressively.

Through the above process, the near single building models are simplified into hierarchical structures. The clients initially receive the coarser data-sets from the server based on the viewpoint. By monitoring the process, the clients can identify the portion of the near scene currently being examined according to the available bandwidth and display resolution. In the adaptive case, the clients request more

detailed data from the server based on the geometry approximation error. The server responds to this request by providing the corresponding bit-stream with increased accuracy in the visible regions. The clients first reconstruct the coarsest models and incorporate them into the data array as they arrive, progressively improving the details of the models being visualized. In this way, it is even possible to refine the models vertex by vertex, respectively. Therefore, the server will be able to adapt the data effort to the clients' capabilities fast and accurately.

Once the visible region changes, message priorities will change and additional data might be required. The clients pre-compute the query region and then they send messages to the server for requesting only the nodes that have not been stored on the clients by specifying the required LODs. Once the clients cannot store the entire multiresolution models, the current non-interested nodes are deleted automatically.

3.2. Image-based transmission for distant buildings

It is observed that buildings far from the viewer appear so blurry that the geometrically modeling method is of limited value. Therefore, we use the image-based method to transfer and render distant buildings. On the server, our image-based method performs hierarchical clustering, cluster merging, model simplification, and hierarchical texturing during preprocessing. It then employs LOD to select the appropriate models to be rendered as view-dependent 3D imposters during the transmission. The process is described in the following subsections.

3.2.1. Building footprint clustering and simplification

To implement building footprint clustering, we divide all building objects into groups, based on the partitions created by the road network. The main reason is that two buildings on opposite sides of a road should not be aggregated. Two steps are needed when different positive buffers are applied to create LOD urban building models. First, different buffer distances are chosen based on the road width in order to achieve different levels of aggregations. If the road width of the study area is acquired, buffer distances (each buffer distance is slightly less than the half of the road width) are set according to the width value. When the width is unknown, it can be determined in the following way: Create positive buffers to the building footprints on both road sides by increasing the buffer distance. When the footprints of the different sides are to be merged, the current buffer distance represents the road width at its narrowest point. Then, positive buffers to the footprints are created using different distances determined by the first step. After the different buffer levels are computed, the Buffer-Tree is built according to the inclusion relation between buffer polygons at adjacent levels. Generally, closer footprints have higher priorities to be clustered.

Figure 1 shows an example for building a Buffer-Tree. Each buffer level is equivalent to one LOD, and a larger buffer distance implies a coarse LOD. In the zero buffer level (buffer polygons are colored in gray in Figure 1), each footprint of an original building is used to create a leaf node. In the non-zero buffer levels (buffer polygons are colored in red and green in Figure 1), each buffer polygon is treated as a non-leaf node, corresponding to the combined footprint of a generalized

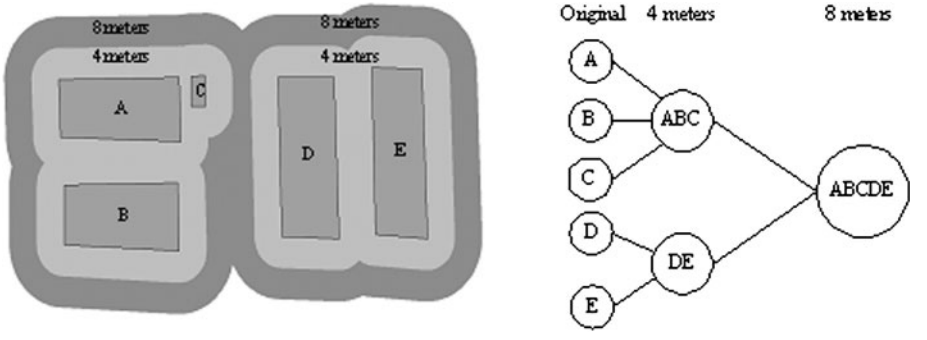

Figure 1. Buffer-Tree generation at different buffer distances.

building model. In the Buffer-Tree, a parent-node can own one or many child-nodes. Since the proposed method allows more than two footprints to be merged simultaneously each time, this approach significantly reduces the depth of the Buffer-Tree and the process has high computation efficiency. Figure 2 shows the footprint aggregation at different buffer distances. Through this procedure, the complexity of the building models is reduced by clustering, thus the speed of this approach is improved greatly.

3.2.2. Render of the clustered buildings

So far, the LOD building models have been established; the building models near to the viewpoint generally have high LODs. The faraway buildings can be rendered by coarse models. We project the error generated by the buildings' simplification onto the screen in real time and use the amount of pixel errors as the criterion for LOD division. The pixel errors should be controlled in a certain range.

In the visualization process, the source of errors which are obviously noticeable can be divided into two main categories. One is related to the height. Although the children nodes have different heights, their father node has only a height value. The other is related with the area. In the merging process, the vacant regions among the buildings will be included in the simplified polygons. For the 2D footprints, the area of the father node is larger than that of all the children nodes. During generating 2.5D models from 2D footprints, the area of the models' roof will increase correspondingly.

When people view the scene from different directions, these two types of errors have different influences on the visual effects. The changes of buildings' roofs cannot easily be noticed when people navigate among the buildings, whereas changes of building heights can be strongly perceived. Particularly, people always pay much attention to the landmark buildings. In this case, the height error should be the main focus. When the navigating path is changed into the bird's eye view looking top-down, the outlines of the roofs are the primary information to human eyes, and the height information becomes less important. In this case we should mainly consider the deformation of the roof contours.

For a non-leaf node, the maximum height $H_{\max}$, and minimum height $H_{\min}$ of its descendants need to be computed through the following procedures:

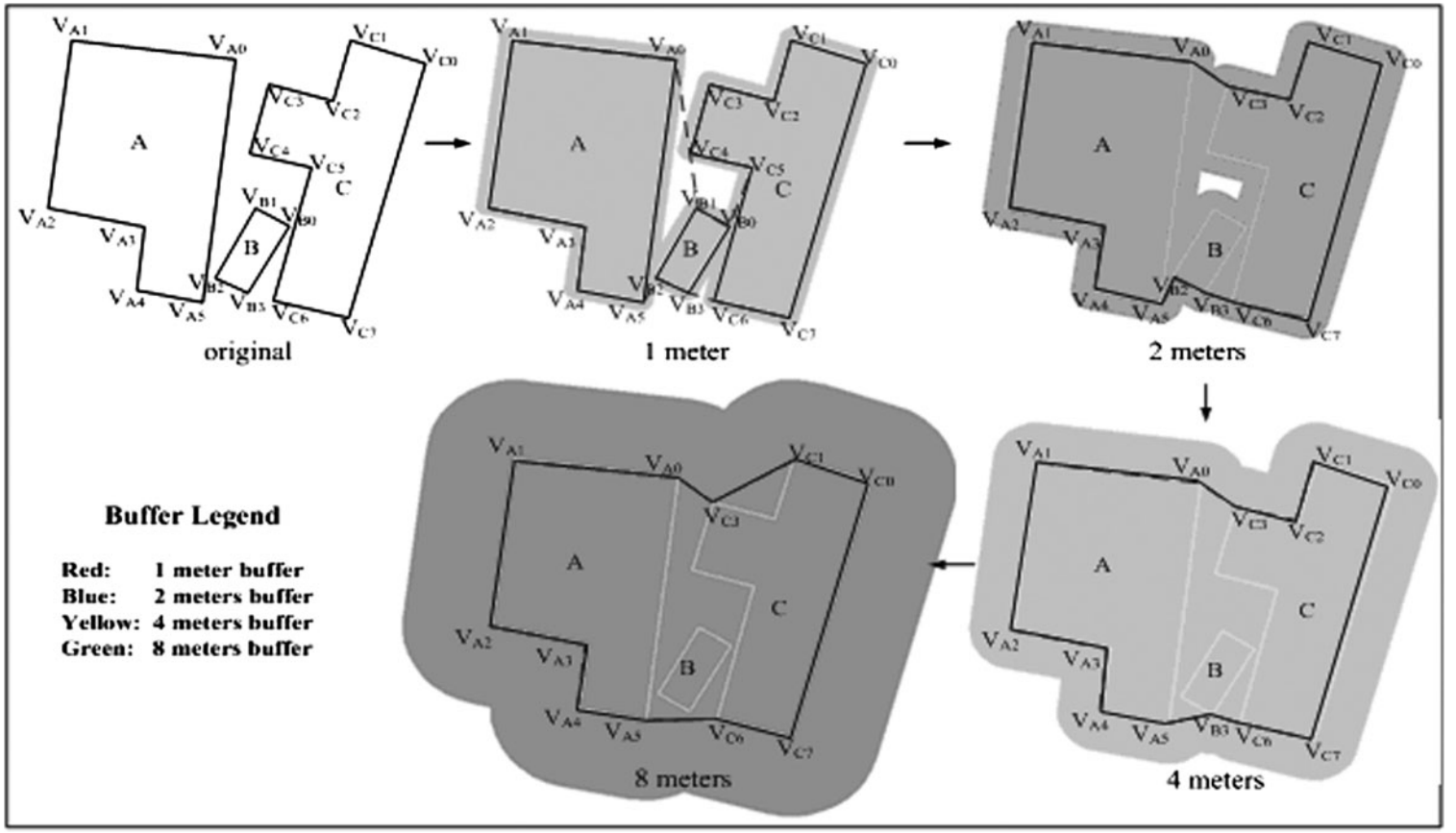

Figure 2. The footprint aggregation at different buffer distances.

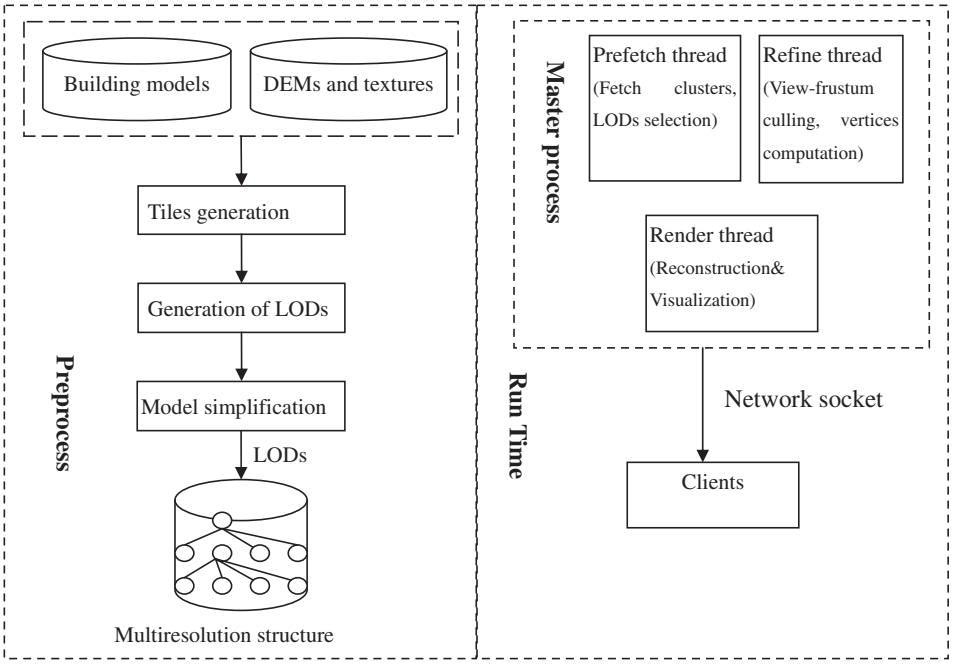

Figure 3. Architecture of the application.

Suppose leaf nodes are located at level n. First, calculate every nodes' $H_{\max}$ and $H_{\min}$ in the level $n-1$. Then calculate these two parameters in the $n-2$ level. For each node in the $n-2$ level, the maximum value $H_{\max}$ of its descendants is selected as this node's $H_{\max}$. Similarly, this node's $H_{\min}$ is the minimum of descendants' $H_{\min}$ values. Define the height difference $\Delta H = H_{\max} - H_{\min}$ as the height difference between the highest and the lowest original buildings which are contained in this node. Repeat this process until level 0 is reached. Obviously, for the leaf node, $\Delta H = 0$.

We define the area difference ΔA as the difference of the area between a node's footprints and the sum of the footprints of the original buildings. Calculation of ΔA is similar with the above process: determine two thresholds δ_H and δ_A; For a non-leaf node, when it is rendered, the height difference in the screen ΔH_{view} should be less than δ_H pixels. Newly added roof area ΔA_{view} should be less than δ_A pixels. ΔH_{view} can be approximated using the method described below.

First, determine the scaling relation between the scene pixels and the unit length in view frustum's near clip plane. The camera's vertical view field is recorded as FOV, and the distance from the camera to the near clip plane is D_{near}, thus the height of the near clip $H_{\text{near}} = D_{\text{near}} \times \tan(FOV) \times 2$. The scene height is H_{scene} pixels, therefore on the near clip plane, per unit length is corresponding to n_{scale} pixels on the screen, $n_{\text{scale}} = H_{\text{scene}}/H_{\text{near}}$. Second, we project ΔH approximately onto the near clip plane. For each node, assume the distance between the node's center and the camera is D_{node}, and the angle between the sight line and the horizontal plane is θ. Then we have $\Delta H_{\text{view}} = \Delta H \times (D_{\text{near}}/D_{\text{node}}) \times \cos(\theta) \times n_{\text{scale}}$.

Similarly, for ΔA_{view}, first approximately obtain its projection on the near clip plane. Then its projection size ΔA_{view} on the screen is $\Delta A_{\text{view}} = \Delta A \times (D_{\text{near}}/D_{\text{node}})^2 \times \sin(\theta) \times n_{\text{scale}}^2$.

During the navigation, because the direction of the sight line and the position of the camera change dynamically, calculate ΔH_{view} and ΔA_{view} in real time. If $\Delta H_{\text{view}} < \delta H$, and $\Delta A_{\text{view}} < \delta A$, render this node. Otherwise, the algorithm does not render this cluster and continue to check its children recursively. It can be seen that this approach not only takes into account the distance between the viewpoint and building models but also considers the geometrical error of the cluster itself. In addition, textures associated with the simplified buildings are also structured based on the method of Chang *et al.*'s (2008).

The server pre-renders the models and stores these images in a repository based on the proposed rendering method. At run time, the client sends a request to display the model at a certain view position and specifies the viewpoint parameters. The server sends back the pre-rendered images. The client calculates and displays the new view by using the image-based rendering with the set of images from the server as imposters for distant buildings.

4. Multi-thread architecture

To compensate for network transmission latencies, we have also used multi-threading techniques to handle the spatial data-sets based on the viewpoints.

4.1. The server

It is an obvious aim to devise an architecture that allows a varying number of clients to send requests simultaneously to the server.

For the server, we propose a multi-process architecture as shown in Figure 3. The data flow is managed by a master process that connects to a network socket, waiting for incoming clients. When a client connects to the server, a new process

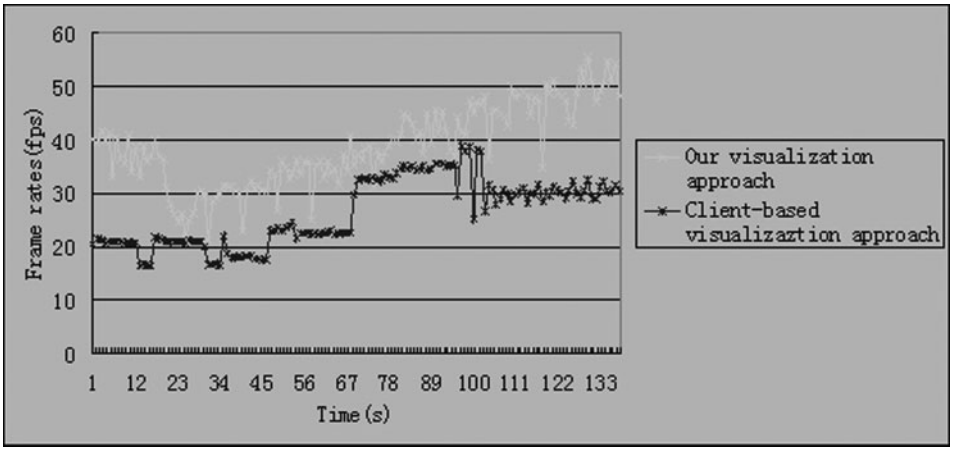

Figure 4. Rendering speed of our method and client-based method.

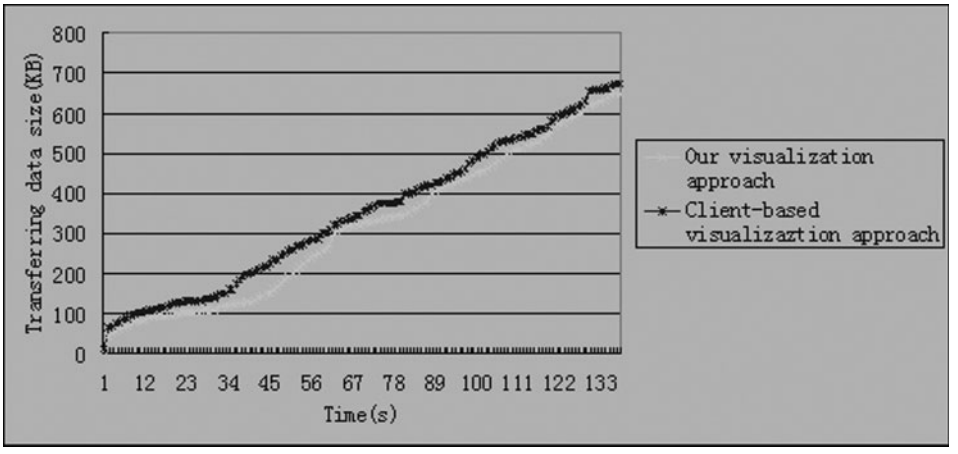

Figure 5. Transferring data rate of our method and client-based method.

is created with an associated socket and a connection established with the client. The child process lives until the connection is closed or the parent process dies.

Processes in the server are independent from each other and do not share any kind of information aside from the read-only city model database. If the building models already exist on the client, it is ignored. Each model gets a priority, which depends on the model's distance to the client's viewpoint. One may argue that the multi-process architecture does not allow two server processes to share in-memory structures, such as the Buffer-Tree used. However, this is not a problem in this case: note that different clients might be navigating over different city zones far from each other; moreover, different clients may move at different speeds and in arbitrary divergent directions. In such a condition, the server does not consume much memory if many clients are connected.

4.2. The client

To compensate for network latencies, our rendering algorithm supports multi-threads including data prediction, refinement, and render running in the back end of

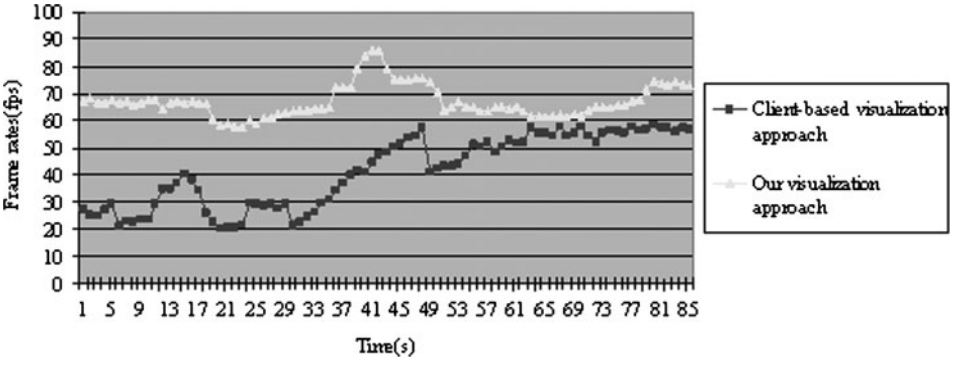

Figure 6. Rendering speed of our method and client-based method.

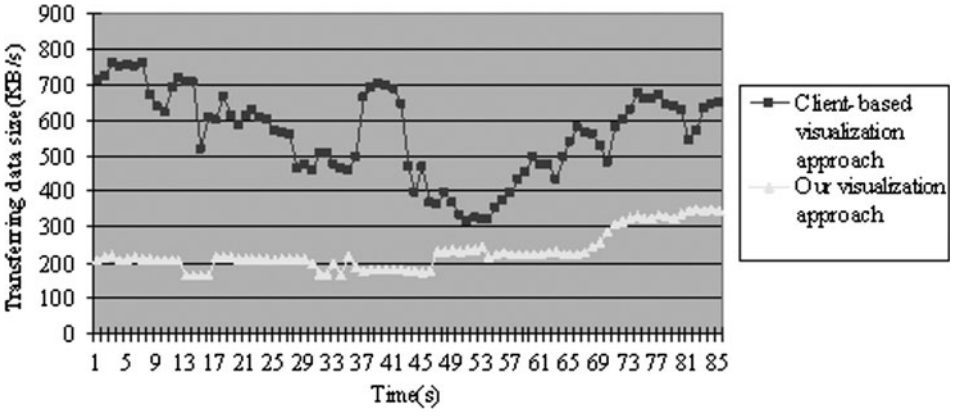

Figure 7. Transferring data rate of our method and client-based method.

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 building models. 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 remain in memory. The refinement operation thread is invoked to compute the visible urban region once the viewpoint changes, responsible for vertex array coordinates update. The render thread implements the multi-resolution rendering. Since the hard disk is the slowest component of the application and new geometries need to be loaded from it, so to reduce latency, the pre-fetch 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 (Figure 3).

5. Experiments

The data transmission and visualization has been implemented in C++ and OpenGL. To investigate the performance of the proposed method, we take the building data-sets of Chaoyang District, Beijing, China, as a test. The data-set

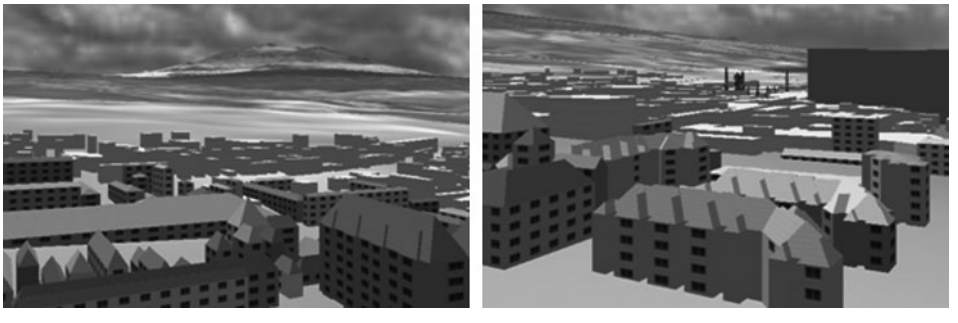

Figure 8. Multiple representations of 3D urban models on the Internet.

contains 35,138 buildings including footprints, roof heights, and corresponding façade textures. These building models have 513,603 footprint vertices. The data size is up to 2GB. We perform the experiments under two network configurations. One is the popular mobile telecommunications technologies: GPRS (2G) and the other is a 1 GBit LAN between the server and the client. We have carried out the experiments on a client with an Intel Core2 Quad Q8200 (2.33 GHz), 2GB main memory, and a GeForce 9600 graphics card with 512 MB video memory. In these network environments, the proposed hybrid rendering technique are compared with the client-based rendering technique.

Figures 4 and 5 illustrate the performance over time of both the hybrid and client-based rendering approaches for the tests. Figure 4 depicts the evolution over time of the frame rate by the two rendering approaches. Figure 5 shows the evolution over time of the downloading measured in Kb.

At the beginning, there is a loading process where both the rendered data and the number of frames per second displayed rapidly changes until reaching certain values that are dependent on the required rendering quality. At the initial load time, the downloading plots are very steep, which suggests the loading process. Once the initial loading is over, both the data rendered and the number of frames per second displayed rapidly change until reaching certain values which depend on the required rendering quality. The data downloaded by our hybrid approach is always smaller than in the client-based approach. Thus, in this case, the use of the images generated on-demand by a remote server does not increase the network usage. In fact, sending images takes less network resources than sending the actual geometry. This shows that the data transfer savings in this hybrid approach scales excellent with regard to the complexity of the landscape.

A 1 GBit LAN between the server and the client was also used to validate our method. Figure 6 illustrates the rendering speed of the two methods. From this, we see that the hybrid rendering approach can constantly render faster than the client-based method. This approach uses a constant number of images to render views, so the rendering speed is very stable shown in Figure 6. Figure 7 illustrates the data transferring rate. From this, we see that our approach can also render views with transfer of less data than the client-based method. We can safely conclude that this hybrid architecture is more applicable for network environments compared to the client-based methods.

Figure 8 shows a part of remote visualization of Beijing city models. From the two figures, we observe that the proposed hybrid approach can keep high visual quality of city models on the Internet.

6. Conclusions

In this study, we have presented a client–server hybrid rendering system for large 3D city models, stored on a remote environment through a network. The approach combines an efficient multi-hierarchical building representation with a novel image-based method, 3D image impostors generated on demand by a remote server. This approach allows transferring complex scenes progressively while keeping high visual quality. All these schemes make the application interactive and operation of multiple collections of city buildings in the networking environments possible. The server can make the data volume adapt to the clients' capabilities precisely. The results

from experimenting with two different Internet environments validated the performance of the proposed approach. From the experiments, we conclude that the proposed approach can greatly reduce transmission time and response time of the spatial data transferred.

Ongoing research is focusing on further improving the multi-resolution controls and adaptive transmission of large terrain models and any objects residing on them.

Acknowledgements

The authors would like to thank the editors and the anonymous reviewers for their valuable comments and suggestions. The study is supported by the National Natural Science Foundation of China (No. 60502008) and the National 863 High-Tech Program of China (No. 2011AA120302).

References

Anders, K.H., 2005. Level of detail generation of 3D building groups by aggregation and typification. *Proceedings of the 22nd international cartographic conference*, 9–16 July, La Coruna, Spain (on CDROM).

Chang, R., *et al*., 2008. Legible simplification of textured urban models. *IEEE Computer Graphics and Applications*, 28 (3), 27–36.

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

Danovaro, E., *et al*., 2006. Level-of-detail for data analysis and exploration: a historical overview and some new perspectives. *Computers & Graphics*, 30 (3), 334–344.

El-Sana, J. and Sokolovsky, N., 2003. View-dependent rendering for large polygonal models over networks. *International Journal of Image and Graphics*, 3 (2), 265–290.

Forberg, A., 2007. Generalization of 3D building data based on a scale-space approach. *ISPRS Journal of Photogrammetry and Remote Sensing*, 62 (2), 104–111.

Ganitseva, J. and Coors, V., 2010. Automatic landmark detection for 3D Urban models. *In*: Thomas Kolbe, König, Gerhard Nagel Claus (Hg.): *Proceedings of International Conference on 3D Geoinformation*. Aachen: Shaker Verlag GmbH (Int. Archives of Photogrammetry, Remote Sensing and Spatial Information Science, Vol. XXXVIII-4/W15), 3–4 November, Berlin, S. 37–43.

Glander, T. and Döllner, J., 2008. Automated cell based generalization of virtual 3D city models with dynamic landmark highlighting. *The 11th ICA workshop on generalization and multiple representation*, 20–21 June, Montpellier, France, The International Cartographic Association.

Glander, T. and Döllner, J., 2009. Abstract representations for interactive visualization of virtual 3D city models. Computers. *Environment and Urban Systems*, 33 (5), 375–387.

Guo, H., Fan, X., and Wang, C., 2009. A digital earth prototype system: DEPS/CAS. *International Journal of Digital Earth*, 2 (1), 3–15.

Guo, H.D., Liu, Z., and Zhu, L.W., 2010. Digital Earth: decadal experiences and some thoughts. *International Journal of Digital Earth*, 3 (1), 31–46.

Huang, B., Jiang, B., and 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., and Kobbelt, L., 2004. View-dependent streaming of progressive meshes. *In*: *2004 International conference on shape modeling and applications (SMI 2004)*, 7–9 June 2004, Genova, Italy. Silver Spring, MD: IEEE Computer Society, 209–220.

Li, Z., 2007. *Algorithmic foundation of multi-scale spatial representation*. Bacon Raton: CRC Press (Taylor & Francis Group), 281.

Lynch, K., 1960. *The image of the city*. Cambridge, MA, USA: The MIT Press.

Meng, L. and Forberg, A., 2007. 3D building generalisation. *In*: W. Mackaness, A. Ruas, and T. Sarjakoski, eds. *Challenges in the portrayal of geographic information*. Amsterdam: Elsevier Science.

Okamoto, Y., Oishi, T., and Ikeuchi, K., 2011. Image-based network rendering of large meshes for cloud computing. *International Journal of Computer Vision*, 94 (1), 12–22.

Pajarola, R. and Widmayer, P., 2001. Virtual geoexploration: concepts and design choices. *International Journal of Computational Geometry and Applications*, 11 (1), 1–14.

Raubal, M. and Winter, S., 2002. Enriching wayfinding instructions with local landmarks. *In*: *GIScience '02: Proceedings of the second international conference on geographic information science*. 25–28 September 2002, London, UK: Springer, 243–259.

Royan, J., Bouville, C., and Gioia, P., 2003. PBTree – a new progressive and hierarchical representation for network-based navigation in urban environments. *Proceedings of the Vision, Modeling, and Visualization Conference (VMV 2003)*, 19–21 November 2003, Germany: Aka GmbH, 299–307.

Royan, J., *et al*., 2006. Network-based visualization of 3D landscapes and city models. *IEEE Computer Graphics and Applications*, 27 (6), 70–79.

Sorrows, M. and Hirtle, S., 1999. The nature of landmark for real and electronic spaces. *In*: C. Freksa and D. Mark, eds. *Spatial Information Theory, International Conference COSIT'99,* 1661, 37–50.

Thiemann, F., 2002. Generalization of 3D building data. *Proceedings of ISPRS commission II symposium ''Geospatial Theory, Processing and. Applications''*, Ottawa, Canada. International Archives of Photogrammetry and Remote Sensing, 34 (Part 4) (CD-Rom).

Thiemann, F. and Sester, M., 2004. Segmentation of buildings for 3D-generalisation. *ICA workshop on generalisation and multiple representation*, Leicester, UK, Available from: http://ica.ign.fr/ [Accessed 23 December 2011]

Winter, S., *et al*., 2008. Landmark hierarchies in context. *Environment and Planning B: Planning and Design*, 35 (3), 381–398.

Zhou, G. and Tan, Z., 2006. Customizing 3D urban visualization via GIS-based internet. *Journal of Urban Planning and Development*, 132 (2), 97–103.