Progressive streaming and massive rendering of 3D city models on web-based virtual globe

Quoc-Dinh Nguyen[1], Mathieu Bredif[2], Didier Richard[3] and Nicolas Paparoditis[4]
Univ. Paris-Est, IGN, ENSG, F-77420 Champs-sur-Marne[4]
Univ. Paris-Est, LASTIG VALILAB, IGN, ENSG, F-77420 Champs-sur-Marne[1,3]
Univ. Paris-Est, LASTIG MATIS, IGN, ENSG, F-94160 Saint-Mandé, France[1,2]
quoc-dinh.nguyen@ign.fr, mathieu.bredif@ign.fr, didier.richard@ign.fr, nicolas.paparoditis@ign.fr

ABSTRACT

The need for the real-time interactive co-visualization of 3D urban environments on a Web-based virtual Globe arises naturally in GIS but it still remains challenging due to the complexity of city models and their huge data sizes which largely overload the computational power and memory capacity of client devices. Especially on the Web, the visualization of city models makes their rendering not real-time because of the lack of content adaptation and progressive data transmission. This paper presents technical solutions for the co-visualization of massive city models in a Web-based virtual globe, allowing navigation over 3D cities on the globe in real-time. The volume of 3D city data, such as building data, does not allow us to render them directly, nor to keep them in the main memory. We propose to use not only a hierarchical presentation of geo-spatial data to create a chunk-based multiple resolution data structure which reduces complexity of the geometry being rendered; but also a view dependent algorithm so that only small subsets of 3D city models are streamed progressively in real-time and kept in client memory to contribute efficiently to the rendered image. Experimental results show that we can navigate over 3D cities on the Globe in real-time.

CCS Concepts

• *Human-centered computing~Web-based interaction* • *Human-centered computing~Geographic visualization* • *Human-centered computing~Visualization toolkits*

Keywords

Virtual Globe; GIS; 3D urban models; massive rendering; progressive streaming; view-dependent LOD control; Chunked-LOD.

1. INTRODUCTION

Virtual globes are known for their ability to render massive real-world terrain, imagery and vector datasets and have brought significant attention to researchers of different domains such as Big Data, Computer Graphics, GIS, etc. The popularity of virtual globes such as Google Earth, NASA WorldWind, Microsoft Bing Maps 3D, and Esri ArcGIS Explorer is almost back-office software.

The internet browsers nowadays show incredible possibilities with HTML5. It makes it possible to use all the power of our device such as sensors, GPS, accelerometer, camera, etc. In addition, the advent of WebGL enables a direct integration of hardware-accelerated 3D graphics into standard web pages without the need of plug-ins. Thus, it provides a way for the creation of new web-based applications that were previously the exclusive domain of the desktop environment. In recent year, many web-based virtual globes, such as Google Maps, Nokia Here, Apple Flyover and Cesium have been developed. Such web-based virtual globes focus on rendering Digital Terrain Model (DTM), Digital Elevation Model (DEM), Imagery and vector datasets but truly 3D city data, such as 3D building, still remains challenging because of its complex data structure and its huge information which makes the co-visualization unworkable with a naive approach.

The main issues are because of the lack of content adaptation (i.e. lack of level of details (LODs)) that makes the systems suffering from network latency (due to the data downloading time and serve multiple users simultaneously); heterogeneous client devices such as mobile phones, tablets and PCs do not work in these cases. Lack of progressive data transmission causes a poor rendering. Data structure is not GPU-friendly creating a bottleneck on the PCI bus. Data loading and data rendering processes are not decoupled which dramatically reduces performance. Also no treatment of rendering artifacts (i.e. popping, z-fitting, jittering, etc) making the co-visualization unpleasant to the eyes.

This paper presents technical solutions for real-time and massive co-visualization of 3D city models on a web-based virtual globe, called iTowns (*). We chose this open-source framework to develop our method because iTowns is the only Web-based virtual globe that is developed under the ThreeJS library supported by a large WebGL/Javascript community. Our paper is organized as follows. In section 2, we present some challenging problems to render 3D urban models in large-scale on globe. In section 3, we explain how we deal with large-scale geospatial data by using tiling system and chunk-based multi-resolution models which support progressive data transmission and view-dependent algorithm. In section 4, we present our modified Chunked-LOD algorithm as a view-dependent algorithm to select and render massively 3D city models in real-time. Experimental results are shown in section 5.

2. Related works

Visualization of data on globe raises many scientific problems. Given the sheer size of the Earth, it requires approximately 2 petabytes of storage for an imagery layer at resolution higher than one meter per pixel. In 2006, Google announce storage of 70 terabytes of compressed imageries that were stored in Bigtable to serve Google Earth and Google Maps [1]. Datasets of this size must be managed with specialized techniques, which is an active area of research.

Various techniques have been presented to face the problem of real-time globe rendering. The key is to locally adapt surface geometric complexity to changing view parameters; the reader is invited to read a good book, written by Cozzi et al [14], to know more about virtual globes. In existing virtual globes, authors focus on developing a view-dependent LOD control for quad-tree structured imagery, DTM and DEM. Nowadays, some very good commercial virtual globes, such as Google Maps, Here, Flyover or even open-source virtual globes such as Cesium or iTowns can

*https://github.com/iTowns/itowns2

render such data in real time. Note that DTM does not take into account the height of buildings or trees and DEM takes into account the relief, but also be called the "canopy" (treetops) which does not have urban furniture and building details. In our case, iTowns can take care of imagery, DTM layers on Globe, and the primary focus of our implementation is towards developing a co-visualization of 3D city models with other data layers of iTowns, so that end-users can view the globe as a whole and zoom into building or street level to see their details.

Fast rendering of 3D city models within web-based applications still remains challenging. Recently, various efforts have been made in order to design progressive delivery of 3D content, for the use with high-performance 3D applications on the web, but not for large-scale geospatial data such as 3D building models. The Khronos Group proposes the glTF format for the transmission of 3D models into WebGL applications. The format is designed in a straightforward manner so that it maps very well to GPU structures on the client side. But the format does not support the progressive transmission of meshes. Other authors try to improve the problem by porting Progressive Meshes (PMs), originally presented by Hoppe [12], to the web [10]. Even that, the lack of view-dependency makes this method still be slow on the web. To deal with rendering complex geometries at interactive framerates, Hoppe [4] proposes to use view-dependent refinement of PMs. The data has hierarchical structure which allows the viewer to remove redundant data and adjust LOD to adapt the mesh complexity according to its contribution to the rendered image. However, the PMs are only working with manifold geometry which is applicable to terrain data but not to 3D city models. In addition, it introduces significant decode time overhead. As alternatives for the PMs, Streaming Meshes, proposed by Lindstrom et al. [11], try to reorder input mesh data in such a way that it can be split into fixed-size memory buffers. This enables a simple progressive transmission of mesh data, and direct upload of downloaded data to GPU memory. In a similar spirit, the POP buffer algorithm [8] sorts the triangle mesh using hierarchy of quantization grids. The triangles can then be grouped according to the precision level where they first appear, which enables the creation of a nested, progressive structure. The data is then saved in SRC format [7] which is a modified version of glTF. The methods focus on progressive mesh transmission; the lack of spatial hierarchical data structure and view-dependent LODs control still make the system suffer from latency; no loading priority is used for data items in this approach. In addition, the file format is not compact and is unusable for our purposes.

3. Chunk-based multi-resolution models

In our case, the 3D city models are reconstructed from Terrestrial and Airborne Lidar point clouds [15]. The models have very high resolution; the amount of data goes to terabytes of storage for a city that does not allow us either to render them directly or to keep them in the main memory. We need two strategies: A chunk-based multiple resolution data structure (CMRs) which reduces complexity of the geometry being rendered and supports progressive data transmission, and a view-dependent out-of-core strategy to filter out as efficiently as possible the data that is not contributing to the rendered image and cope efficiently with the insufficient amount of GPU and CPU memory. Our method must scale well with our large-scale geospatial datasets; enabling a progressive transmission of mesh data; reducing loading time with data compression; eliminating decode time by decoupling render thread and decode thread; the data structure must be GPU friendly so that it can be uploaded directly to GPU after decompression.

SIGSPATIAL'16, October 31 - November 03, 2016, Burlingame, CA, USA
ACM 978-1-4503-4589-7/16/10.
http://dx.doi.org/10.1145/2996913.2997008

Dealing with city wide 3D models. Difficulties arise when rendering large-scale 3D models of a city because simplification methods start from a detailed mesh and the city models contain huge information that does not fit to main memory. Various techniques have been presented to face the problem of huge mesh simplification: Out-of-core construction using memory insensitive simplification, proposed by Lindstrom [3], to construct a multiresolution hierarchy; External memory management proposed by Cignoni et al [2] to construct an octree of huge meshes. These methods seem to be very interesting, they can deal with a huge mesh of hundred million of triangles, but it still does not work with our large-scale geospatial data which has quadrillion of triangles and may take terabytes of storage for whole city.
To deal with that, we borrow an idea from traditional approach of imagery, the city models are partitioned into tiles (i.e. squared mesh block). By partitioning the data into tiles, we can address main memory constraints, but the tile data were still heavily GPU /CPU bound. Even when we apply LOD on tiles, it is unable to generate model updates at full GPU speed and to efficiently communicate them to the graphic card, because of mesh size at high LOD. A solution is to depart from multiple LOD meshes and to adopt a chunk-based hierarchical data structure; each chunk is a set of triangles, a connected part of the mesh, from which our view-dependent Chunked-LOD algorithm can efficiently stream and render data. In addition, with a tiling system, we can easily apply horizon culling to eliminate tiles which are occluded by the horizon.

Chunk-based multi-resolution model for tile. Some parts of tile may be occluded or outside the view frustum and thus do not contribute to the rendered image. In addition, some regions of a tile may lie closer to the viewer than others; so separated parts of a tile need to be rendered at different levels of details. This problem can be addressed by creating a hierarchy of meshes within a tile. In [2][3], authors propose to use an octree or a kd-tree to partition a mesh into multi resolution fragments. But their multiresolution structures make the CPU a bottleneck because they take decisions at the triangle/vertex primitive level. This kind of approach is thus not able to choose what has to be rendered fast enough.

To overcome this bottleneck, we reuse the ideas proposed in [5][6], where the granularity is moved from individual triangles to chunks of triangles. We use a Kd-tree to arrange mesh fragments (chunks) at different resolutions. The chunk is optimized in a GPU friendly fashion with a triangular-stripe layout and is thus ready to send to GPU memory on demand. The root node if each tile thus contains a drastically simplified version of its original mesh. Each child node contains a subset of its parent, where each subset is more detailed than its parent but covers a smaller spatial extent. The union of the meshes borne by leaves of Kd-tree (maximum depth) encodes the original tile mesh at full resolution. In rendering, GPU only needs to consider each chunk rather than considering individual triangles, as is required in state of the art algorithms. A chunk in the distance may be rendered with less geometry and lower resolution textures if it were close to the viewer. In this way, the amount of processing that needs to be done by the CPU/GPU is greatly reduced. CMRs can also reduce the quantity of data sent to the GPU over the system bus

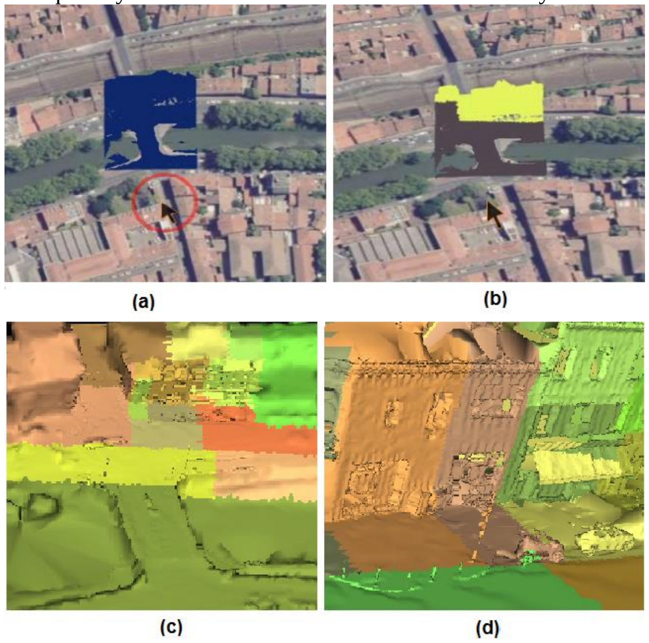

Figure 1: A tile mesh of 100x100 meters, containing 350K faces and 256K vertices. Figure (a) illustrates root node that is drastically simplified version of tile. Figure (b) shows second level of tree when camera approaches and (c) (d) shows its other levels when camera is close to the building façade, we can see a more detailed mesh without any cracking issue. The colors identify the chunks.

Cracking and popping artifact. Chunk-based multi resolution produces popping artifacts when switching between different LODs. To enforce continuity of the surface along the boundary of the chunks at different resolutions, many techniques are proposed such as marking, stitches and geo-morphing [6]. Among them, we found that the geo-morph technique works well; it allows eliminating popping artifacts between different LODs. The technique interpolates smoothly geometries from different LODs. An example of geo-morph is illustrated in figure 1 (d)

Pre-computed viewing parameters. The viewing parameters (such as the geometric error, the chunk's bounding volume, horizon culling points, pivot, etc) are used by the Chunked-LOD algorithm to eliminate redundant data. These parameters have a high computational cost. In our work, such parameters are pre-computed so that the viewer only needs to load them as metadata without any additional computation.

Progressive data streaming format. We want to save our tile in a file but we do not want to send all data within a single batch as our tile mesh has a hierarchical data structure. We found the Nexus format (**) which is very well designed for multiple resolution meshes. We extend this format for our geo-spatial data by adding the parameters that will be used by our Chunked-LOD algorithm (HC points, pivots, etc).

4. Massive rendering on Globe

Considering that 3D models for a city are often measured in terabytes, it comes as no surprise that the entire dataset cannot be in client memory (GPU/CPU's) all at once. No web-based application can cope with this size, which demands us to cut the dataset into tiles and construct CRMs for each tile. This hierarchical data structure motivates the use of out-of-core rendering algorithms. The idea is to visualize only tiles which have compatible viewing parameters (view frustum, HC, SSE). By using view parameters, new portions (chunks) of the tile mesh are brought into system memory, and old portions are removed, ideally without stuttering rendering. This allows us to keep only small subset chunks in system memory. The rest resides in a file in secondary storage on network server.

In this paper, we do not combine our tiles into a unique tree for a city or all cities on the Earth. Instead, we manage multiple trees to show a city on the globe, this allows us to easily use horizon culling (HC) technique to eliminate tiles which are occluded by the globe shape.

Chunked-LOD algorithm. We extended a view-dependent algorithm, called Chunked LOD, originally proposed by Ulrich [13], for massive terrain rendering. Unlike Ulrich's method, our Chunked-LOD algorithm operates on tiles and also on chunks. In addition, we use an additional culling condition to eliminate tiles which are not contributing to the rendered image. Our algorithm is structured as in figure 2.

Our Chunked-LOD algorithm operates first on tiles to verify which tiles are visible from the camera, checking two conditions: Frustum and Horizon. Thanks to the pre-computation of viewing parameters, we do not need to load chunk data to estimate these parameters. The viewing parameters are saved in file header and are only loaded the first time the tile appears to be visible from the camera. For each visible tile, we start at root node, according to screen space error (SSE), if the node has sufficient details for the scene, it is rendered. Otherwise, the node is refined, meaning that its children are considered for rendering instead. This process continues recursively until the entire tile is rendered.

```
For each tile:
   If current tile is in frustum and satisfy HC
      Load viewing parameters
      If chunk meets SSE requirement
         Add chunk to render list
      Else
         Recurse on visible child chunks
```

Figure 2: View-dependent Chunked-LOD algorithm

Unlike original Chunked-LOD algorithm which needs data to be available in client memory to compute viewing parameters, our method runs very fast because the viewing parameters are pre-computed. Only relevant chunks, which are added to the render list, must be downloaded from the server.

Screen space error (SSE). We determine the required chunk LOD with SSE which measures the number of pixels of difference that would result by rendering a lower-detail version of the chunk rather than its higher-detail version. The SSE is upperbounded by dividing the geometric error by the distance from the viewpoint to the bounding sphere. If the bounding sphere is outside of the camera's frustum, the SSE is set to zero, if the viewpoint is inside the bounding sphere, the SSE is infinite

Tile horizon culling. When it comes to rendering a city on a globe, we need to consider the Earth itself as an occluder especially when the viewer is close and parallel to terrain surface, horizon culling really eliminates many occluded tiles (***).

**http://vcg.isti.cnr.it/nexus/

*** https://cesiumjs.org/2013/04/25/Horizon-culling/

5. Results

Our system is designed for high resolution data as the one illustrated in figure 1. The tile mesh is reconstructed from Terrestrial and Airborne Lidar point clouds and contains 350K faces and 256K vertices within a 100x100 meters extent. At this time, we do not have the data in large-scale, for a city by example, but this data can go to terabytes of storage for a whole city. As we can see in the figure 1, thank to Chunked-LOD algorithm, only chunks which are in the view frustum and satisfy SSE will be extracted and be streamed progressively to web browser. The parts of tile mesh which are outside of camera frustum stay on the server side in the same file. During navigation, new portions of the mesh are brought into system memory and old portions are. Precomputed chunks are also compressed and organized in GPU friendly fashion; so that they can be sent and updated easily to GPU memory with zero copy GPU uploads. Decompression can be done in rendering thread but doing that causes severe stalls, making the globe unusable. Instead, chunks are loaded and decompressed in separate threads in our Chunked-LOD program.

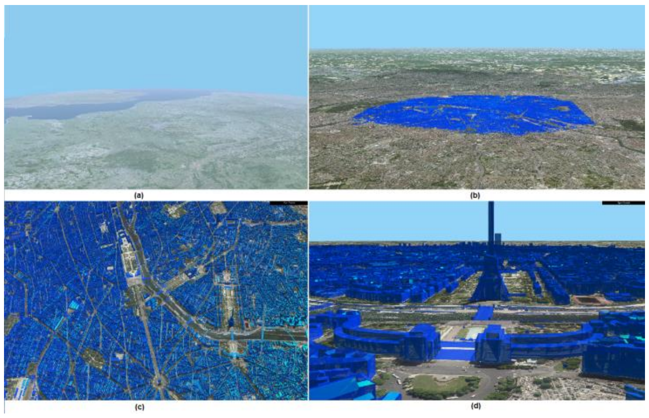

Figure 3: Co-visualization of Bati3D with iTowns's virtual globe

The data illustrated in figure 1 is not available at this time for a whole city. We test our system again large-scale problem with the Bati3D, 3D city models produced by IGN. This database contains 3D building models of some French cities. Figure 3 shows how our proposed method can visualize Bati3D of the entire Paris city within the iTowns virtual globe. This database is partitioned into squared tiles of 500x500 meters. For a whole city such as Paris, we have around 400 tiles. Each tile contains approximately 100K faces and vertices. When camera is far from the terrain (as illustrated in figure 3 (a)), the view encompasses the entire globe, the entire terrain is inside the frustum and thus nothing is culled. SSE becomes very important, at such an altitude, any building details are visible. Thanks to SSE, no building is selected to render list (figure 3 (a)), only imagery and DTM are rendered on the globe. Figure 3 (b) shows the footprint of 3D building model; at this level, only simplified building mesh of the root node is downloaded and rendered. When zoomed in, however, our Chunked-LOD method eliminates a large percentage of the tiles and chunks that should not be rendered. Figure 3 (c) illustrates more building details when camera is closer to them.

Conclusions and future works: We proposed some technical solutions to co-visualize 3D city models on Web-based virtual globe. Experimental results show that we can navigate over 3D cities on the globe in real-time. For future works, the approach will be extended to multiple directions, including the support of textured meshes and heterogeneous tile and chunk hierarchies.

6. REFERENCES

[1] Fay Chang, Jerey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A.Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E.Gruber. Bigtable: A Distributed Storage System for Structured Data."In Proceedings of the 7th Conference on USENIX Symposium on OperatingSystems Design and Implementation, 7, 7, pp. 205

[2] Paolo Cignoni, Claudio Montani, Claudio Rocchini, and Roberto Scopigno. 2003. External Memory Management and Simplification of Huge Meshes. *IEEE Transactions on Visualization and Computer Graphics* 9, 4 (October 2003), 525-537.

[3] Peter Lindstrom. 2003. Out-of-core construction and visualization of multiresolution surfaces. In *Proceedings of the 2003 symposium on Interactive 3D graphics* (I3D '03). ACM, New York, NY, USA, 93-102.

[4] Hugues Hoppe. 1997. View-dependent refinement of progressive meshes. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (SIGGRAPH '97). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 189-198.

[5] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio and R. Scopigno, "Batched multi triangulation," *VIS 05. IEEE Visualization, 2005.*, 2005, pp. 207-214. doi: 10.1109/VISUAL.2005.1532797

[6] S. E. Yoon, B. Salomon, R. Gayle and D. Manocha, "Quick-VDR: out-of-core view-dependent rendering of gigantic models," in IEEE Transactions on Visualization and Computer Graphics, vol. 11, no. 4, pp. 369-382, July-Aug. 2005.doi: 10.1109/TVCG.2005.64.

[7] Max Limper, Maik Thöner, Johannes Behr, and Dieter W. Fellner. 2014. SRC - a streamable format for generalized web-based 3D data transmission. In *Proceedings of the 19th International ACM Conference on 3D Web Technologies* (Web3D '14). ACM, New York, NY, USA, 35-43.

[8] M. Limper, Y. Jung, J. Behr, M. Alexa: "The POP Buffer: Rapid Progressive Clustering by Geometry Quantization", Computer Graphics Forum (Proceedings of Pacific Graphics 2013).

[9] Adrien Maglo, Ho Lee, Guillaume Lavoué, Christophe Mouton, Céline Hudelot, and Florent Dupont. 2010. Remote scientific visualization of progressive 3D meshes with X3D. In *Proceedings of the 15th International Conference on Web 3D Technology* (Web3D '10). ACM, New York, NY, USA, 109-116.

[10] Guillaume Lavoué, Laurent Chevalier, and Florent Dupont. 2013. Streaming compressed 3D data on the web using JavaScript and WebGL. In Proceedings of the 18th International Conference on 3D Web Technology (Web3D '13). ACM, New York, NY, USA, 19-27.

[11] M. Isenburg and P. Lindstrom, "Streaming meshes," VIS 05. IEEE Visualization, 2005., 2005, pp. 231-238. doi: 10.1109/VISUAL.2005.1532800

[12] Hugues Hoppe. 1996. Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (SIGGRAPH '96). ACM, New York, NY, USA, 99-108.

[13] Thatcher Ulrich. Rendering Massive Terrains using Chunked Level of Detail Control, cours in Super-size it! Scaling up to Massive Virtual Worlds, SIGGRAPH, 2002.

[14] Patrick Cozzi and Kevin Ring. 2011. *3D Engine Design for Virtual Globes* (1st ed.). A. K. Peters, Ltd., Natick, MA, USA.

[15] N. Paparoditis, J.-P. Papelard, B. Cannelle, A. Devaux, B. Soheilian, N. David, E. Houzay. Stereopolis II: A multi-purpose and multi-sensor 3D mobile mapping system for street visualisation and 3D metrology. *Revue Française de Photogrammétrie et de Télédétection 200: 69-79*, October 2012