OUT-OF-CORE CONSTRUCTION AND 3D VISUALIZATION OF LEVEL-OF-DETAIL TERRAINS POPULATED WITH LARGE COLLECTION OF HETEROGENEOUS OBJECTS

Anupam Agrawal, M. Radhakrishna
Indian Institute of Information Technology (Deemed University), Deoghat, Jhalwa, Allahabad – 211 012, India

R.C. Joshi
Dept. of Electronics & Computer Engg., Indian Institute of Technology, Roorkee – 247 667, India

Keywords: Terrain visualization, Out-of-core algorithms, Multiresolution modeling, View-dependent refinement.

Abstract: Interactive visualization of very large-scale terrain data in scientific visualization, GIS or simulation and training applications is a hard problem. The grid digital terrain elevation and texture data are not only too large to be rendered in real-time but also exceed physical main memory capacity. Therefore out-of-core management of digital terrain data is an essential requirement. Further to bring photorealism in visualization, it is required to place multiple collections of man-made objects such as buildings, lampposts etc. as well as natural objects such as trees, grass etc. on top of the terrain surface. In this paper we have proposed an integrated approach for effective out-of-core visualization of terrains populated with large collection of static heterogeneous objects. We have developed an efficient tile-based out-of-core view-dependent Level of Detail (LOD) mesh simplification algorithm for real-time rendering of large terrains. Instead of manipulating individual triangles, the algorithm operates on clusters of geometry called blocks of aggregate triangles. Hence the amount of work CPU must perform is greatly reduced. The formation of long triangle strips for LOD blocks also solves the CPU-to-Card bandwidth problem. The tile-based multiresolution terrain geometry framework has been extended to support large satellite or aerial imagery textures. To display large collection of objects over the terrain while maintaining the real-time frame rate, an efficient object handling method has been proposed using paging technique and object instantiation. User is allowed to control the objects locations, scales and orientations. The algorithms have been implemented using Visual C++ and OpenGL 3D API and successfully tested on different real-world height maps and satellite phototextures of sizes upto 16K*16K coupled with thousands of static objects on PCs.

1 INTRODUCTION

There is a growing demand for high quality interactive visual simulations in the field of landscape visualization. However, interactive visualization of large-scale terrain data brings up a wealth of problems. Because the grid digital terrain models are not only too large to be rendered in real-time but also exceed physical main memory capacity, traditional in-memory multiresolution meshing and rendering techniques do not provide a sufficient solution. Relying only on virtual memory and the operating system's paging mechanism does not sufficiently take into account spatial as well as level-of-detail relations in a multi-resolution triangulation framework. For rendering large terrain from out-of-core, the multi-resolution data structure and rendering algorithm themselves must provide an efficient paging of LOD data from disk. The same argument holds true for rendering large collection of static heterogeneous objects such as buildings, trees etc. on top of the terrain surface.

To achieve real-time rendering without sacrificing accuracy, several aspects have to be considered. On one hand, to exploit the full performance of the current Graphics Processing Units (GPUs) hardware, transmission of large data chunks is advantageous. On the other hand, no unnecessary data should be submitted for rendering, since bandwidth and I/O are often the bottleneck of

Agrawal A., Radhakrishna M. and Joshi R. (2006).
OUT-OF-CORE CONSTRUCTION AND 3D VISUALIZATION OF LEVEL-OF-DETAIL TERRAINS POPULATED WITH LARGE COLLECTION OF HETEROGENEOUS OBJECTS.
In *Proceedings of the First International Conference on Computer Graphics Theory and Applications*, pages 429-435
DOI: 10.5220/0001358404290435

current graphics systems. Furthermore, with the growing GPU power the management of fine-grained LODs on the CPU as done by traditional algorithms (Lindstrom, 1996, and Duchaineau, 1997) becomes more and more limiting factor, and in many rendering applications the GPU is not working at full capacity.

This paper discusses the methodology and implementation aspects of our research work to improve the quality and speed of rendering of large terrains with objects on general-purpose desktop PCs. Our block-based dynamic LOD terrain-rendering software, TREND, considers above facts using 3D rendering hardware and minimizes the CPU overhead. We have proposed an integrated approach for effective out-of-core visualization of terrains populated with large collection of discrete, static heterogeneous objects. User is allowed to control the objects locations, scales and orientations. The object instantiation scheme reduces the memory overhead to a large extent.

2 RELATED WORK

External memory algorithms (Vitter, 2001), also known as out-of-core algorithms, address issues related to the hierarchical nature of the memory structure of modern computers (fast cache, main memory, hard disk etc.). Managing and making the best use of the memory structure is important when dealing with large data structures that do not fit in the main memory of a single computer. In most terrain visualization systems, two approaches are prevailing for external memory handling. In the first approach (Dollner, 2000, and Lindstrom, 2001) the multiresolution terrain triangulation hierarchy is linearized into an array and a memory-mapped file mechanism (supported by the operating system) is used to provide out-of-core access. The second approach (Reddy, 1999, and Pajarola, 1998) is to split the terrain into large rectangular tiles of varying resolution that are paged in on demand. The main drawbacks of the first approach are that the terrain data is only clustered on disk with respect to the linearization of the triangulation hierarchy and that the storage cost is comparatively high. We have adopted the second approach for out-of-core data management for terrain topography as well as for the objects.

Many mesh simplification and multiresolution triangulation methods have been developed over the last decade. Due to space constraint, we refer to the literature for overviews on general mesh simplification and multiresolution modelling (Cignoni, 1998, and Luebke, 2001). We have chosen regular hierarchical structure to represent terrain (stored as height map) as it allows fast collision detection between moving objects (including camera) and the terrain. It also supports use of efficient hierarchical data structures for fast and easy view frustum culling.

Relatively less work has been reported in literature on object management over multiresolution terrain. Szenberg et al. (Szenberg, 1997) describe a method of terrain visualization with point-location based objects such as houses, transmission poles etc. The visualization scheme for terrain height field is not based on multiresolution modelling but combines the Z-buffer with the Floating Horizon algorithm. Further it has been applied to limited sized terrain data (512*512 size) only. Douglass et al. (Douglass, 1999) describe a bottom-up LOD height-field rendering scheme by placing building-like objects over the terrain. In contrast to a top-down LOD approach, a bottom-up approach necessitates the entire model being available at the first step and therefore has higher memory and computational demands (Luebke, 2003). In this paper, we have proposed a new object management approach coupled with our block-based multiresolution LOD terrain modelling approach. It employs an efficient object-paging scheme, which smoothly adapts to our tile-based organization of geometry and texture data for out-of-core data management.

3 THE TILE BASED MULTI-RESOLUTION FRAMEWORK

We have developed a view-dependent dynamic block-based LOD modelling for mesh simplification and using tiled geospecific texture, to display the details of the high-resolution satellite imagery in real-time rendering (Agrawal, 2004a). The terrain geometry and texture data are organized in titles of size 257*257 and 256*256 respectively as shown in Figure 1. One pixel overlap is kept between adjacent geometry tiles to ensure proper stitching of tiles. At any instance of time, only nine tiles are kept in main memory. The viewer position is always assumed to be inside the centre tile. The algorithm efficiently handles out-of-core data by dynamic paging of terrain tiles between secondary storage and main memory. Each geometry tile data is organized with a quadtree with leaves corresponding to patches or

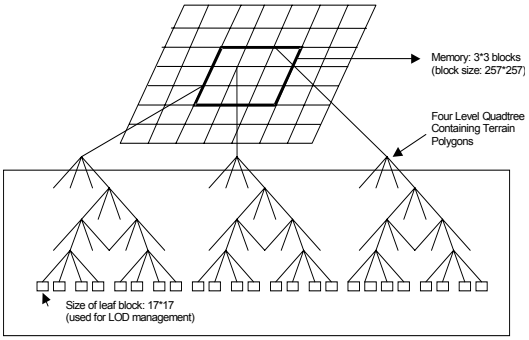

Figure 1: Organization of Terrain Geometry Data.

blocks of size 17*17 (the size decided after experimentation) to speed up the view-frustum culling.

Multiresolution pyramid representation is used to define each terrain block. Figure 2 shows the four pyramid levels of the height map block of size 17*17. Considering the multiresolution representation for each patch, the algorithm employs a variable screen-space threshold (τ) to limit the maximum error of the projected image considering the terrain complexity, viewer distance and viewing direction as the viewer navigates the terrain. The algorithm pre-computes a look-up table at terrain tile load time to decide the tessellation level of each block within view-frustum based on position of the camera from the block (De Boer, 2000). In the approach, a group of vertices are considered instead of single vertex for deciding whether to remove them or not. Hence CPU requirements are many times lower as compared to the LOD mesh simplification algorithms, which work on individual vertices of the height field.

It is important to note that in a view-dependent framework, the resolution of adjacent patches might change at every frame. Hence, cracks occur on borders of adjacent patches of different levels of detail. In Figure 3 (a), the circle shows the position of crack in tessellation with level difference one (right side patch is shown partially). Figure 3 (b) shows the modified geometry to remove the cracks where the dashed edges are excluded in triangulation and the bold edges are included. Similar procedure is followed to eliminate cracks when level difference between adjacent patches is two or three. Image draping over 3D mesh geometry is performed using texture mipmapping. The algorithm also handles the problem of texture seams between adjacent texture tiles (Agrawal, 2004b).

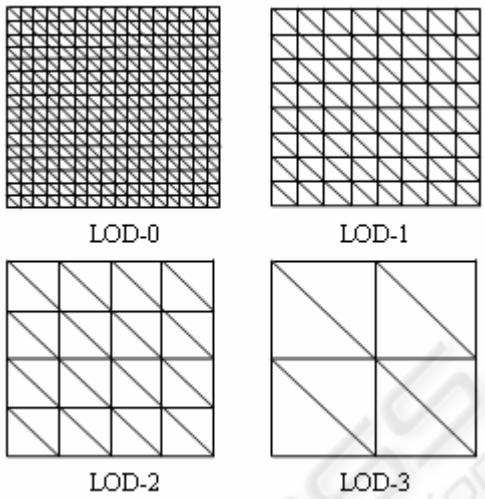

Figure 2: Multiresolution Modelling of Height Map.

To exploit the full performance of current GPUs hardware, transmission of large data chunks is advantageous. Graphics rendering can be accelerated through compact representations of polygonal meshes using data structures such as triangle strips and triangle fans. Using triangle strip primitive, it is possible to form a longer length of connected

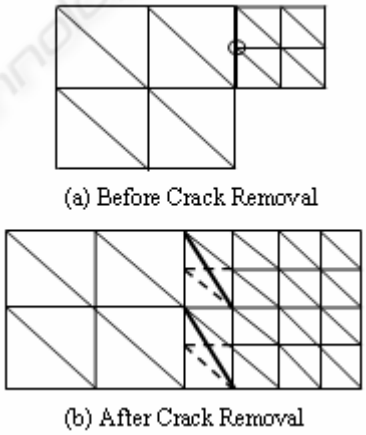

Figure 3: Removing Cracks between Adjacent Patches.

triangles as compared to triangle fan. Generating long triangle strips efficiently solves the CPU-to-card bandwidth problem and avoids redundant 3D vertex transformation and lighting (T&L) calculations. However in view-dependent meshing methods the underlying mesh is in a constant state of flux between view positions. This poses a significant hurdle to construct long triangle strips. Our triangle strip generation scheme for view-dependent dynamic multiresolution terrain shows significant improvement in rendering speed as compared to individual triangle-based and triangle-fan based rendering schemes (Agrawal, 2005). The snapshot of

the triangulated height map is given in Figure 4 using triangle-strip primitive.

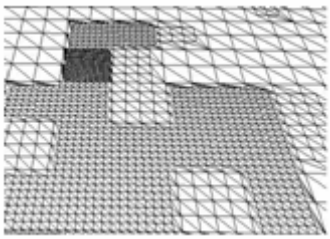

Figure 4: Wireframe View of Terrain LOD Geometry.

4 OUT-OF-CORE VISUALIZATION OF OBJECTS

We may divide the point-location objects into two categories. The first one is simple objects, those having simple geometry and can be drawn with the help of OpenGL primitive functions. These objects do not need to be loaded into memory. Examples of simple objects include sky-scrapper buildings drawn using elongated cube with texture mapping over its exposed faces and also objects created using billboarding technique. The second category is of complex objects having complex geometry and large number of triangles. These objects are required to be loaded into main memory containing their vertices and topology information. Examples of such objects include complex 3D geometrical models of buildings, trees etc. The software has the provision to import such complex structures from .dxf format and convert into native .mesh files obtained after selecting relevant only information. Complex objects may consume substantial memory as well as drawing time and may severely affect the rendering performance. So we usually prefer to use simple objects to populate the terrain.

While having a walkthrough over the 3D terrain, user cannot see all the objects at once. Thus there is no need of keeping all of them in memory and render them. Our objective is to deal with large number of objects over the terrain. The number of objects is not constant throughout the navigation process; also more objects can be edited, added, or even deleted from the objects list. To enable real-time rendering and interactive communication between user and objects, an efficient method of object handling has been proposed and implemented using dynamic paging technique and object instantiation.

4.1 Placing Multiple Instances of Objects over Terrain

The first task has been the deployment of various building-like objects on the terrain. The various types of buildings and houses being rendered on the terrain would give a look of a good human settlement. The software has the provision for various types of buildings like clock-towers, skyscrapers, houses etc to be shown on the terrain. The buildings could be designed using OpenGL 3D API itself or using AutoCAD package (in DXF-3DFACE format). The software helps the user to place a building anywhere on the terrain by pointing at the scanned georeferenced map or image in the background inside a 2D window after selecting the specific building model. Once the buildings are deployed on the terrain, the software permits the user to edit the objects using all the conventional edit features for the objects such as scale change etc.

An object may have its multiple instances with possibly different scaling factors. If an object is already been loaded into memory, on its subsequent occurrences simply the pointer of object memory is returned for further processing and its counter is incremented by one. Thus the multiple loading of same objects can be avoided.

For many objects and a few special effects such as storm etc. we have also used the technique of billboarding which is sometimes of great use as the loss in framerate is quite negligible and it displays the 2D images just like 3D objects. Billboarding is a technique that adjusts an object's orientation so that it "faces" some target, usually the camera. Billboarding can be used to cut back on the number of polygons required to model a scene by replacing geometry with an impostor texture. For example this technique has been used to create trees, lampposts, signposts etc. This particular technique can be used in these cases when finer detail is not required about the object. Here we have used small size .bmp or .tga images to use as textures, which take less space and hence help us in achieving a better frame rate i.e. faster rendering of the terrain with objects. Only object's memory reference and new scaling factors are kept in memory for multiple instances of same object.

4.2 Paging and Display of Objects

When user opens the model layer consisting of objects, a message is sent to Objects class to open the particular file, followed by loading of names and locations of all the objects, which are there in file. These all objects (i.e. their geometry & texture) are not loaded into memory, but the objects of same tiles are grouped together, so that objects of current nine tiles can be loaded into memory. The software internally manages a dynamic data structure to store tile-wise objects details (without their geometry & topology information). The geometry & topology information of only those complex objects are kept in main memory, which are inside current active nine tiles.

When new tiles are loaded and old are deleted, 'AddModel' function is called to add new models and 'RemoveModel' is called to remove old models of corresponding tiles. The 'AddModel' function is called prior to 'RemoveModel' function to optimize the time. The model is deleted from the linked-list in memory only when there is no instance of same model in current scene. Otherwise counter is incremented or decremented accordingly. Objects paging helps in out-of-core management of large objects data on secondary storage.

5 RESULTS AND DISCUSSION

The software TREND has been tested with 2K*4K terrain raster dataset of Grand Canyon and 16K*16K terrain data set of Puget Sound area obtained from Georgia Institute of Technology website. We have also generated the height map of Dehradun (India) area using digitised contours on Survey of India (SOI), India supplied topographic map. The corresponding geo-referenced IRS-1D FCC Satellite imagery has been used for image draping.

The images in Figure 5 and Figure 6 show the Height map (DEM) of Puget Sound area and corresponding False Color Composite Satellite imagery respectively.

Figure 7 and Figure 8 show the level of detail 3D wiremesh view and corresponding phototextured view respectively. Figure 9(a) shows a view with point-location based simple objects (OpenGL and Billboard drawn objects). A complex object (a building designed using AutoCAD) is shown over the terrain in Figure 9 (b).

The performance of the software has been evaluated on a Pentium IV 2.4 GHz computer with 512MB RAM and Intel 82865G onboard Graphics Controller on 865GL motherboard. The performance of the algorithm on raster data is independent of size of terrain data as with the tiles indexing scheme, the

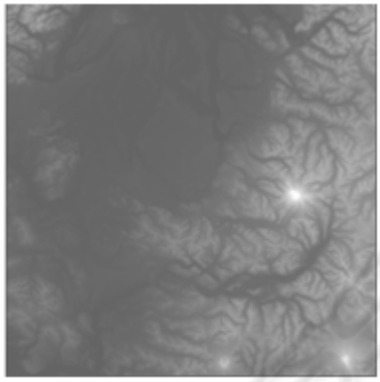

Figure 5: DEM (Height Map) of Puget - Sound area (size 16K*16K).

algorithm only keeps nine tiles active in main memory. The organization of the terrain data in tiles of defined size is required to be done only once on the same data set. For raster data, the number of frames rendered per second mainly depends on the complexity of the terrain (roughness) under the view-frustum and the user defined image quality metric (τ). The graphs in Figure 10 show the effect of τ on rendering performance. Table 1 shows performance analysis of the Adaptive LOD Algorithm for 3D visualization of the raster data set.

The results of testing the same adaptive LOD algorithm using triangle strip (with indexed vertex array) in conjunction with object management algorithm are shown in Table 2. The value of τ is kept 4.0 to obtain various results as shown in Table 1 and Table 2.

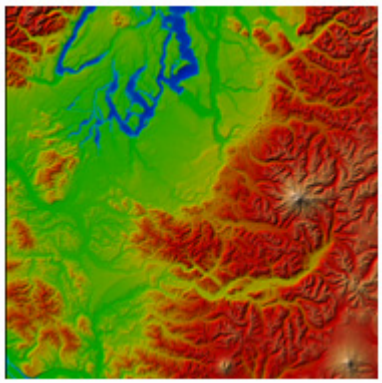

Figure 6: FCC Satellite Imagery of Puget - Sound area (size 16K*16K).

6 CONCLUSION AND FUTURE WORK

This paper discusses the methodology and implementation aspects of our research work to improve the quality and speed of rendering of large terrains on general-purpose desktop PCs. The proposed LOD algorithm for raster data uses a compact and efficient multi-resolution grid representation of height fields and employs a variable screen-space threshold to limit the maximum error of the projected image. The method is different from the individual triangle-based LOD algorithms and is optimised for modern, consumer 3D graphics cards and minimizes CPU usage during rendering. It is augmented with out-of-core visualization of large height geometry and texture terrain data. To display large collection of point-location based static objects over the terrain while maintaining the real-time frame rate, an efficient object handling method has been proposed using paging technique and object instantiation. Objects include different kinds of buildings, trees etc. User is allowed to control the objects locations, scales and orientations.

As a next step to further improve the rendering performance and quality of visualization, we are currently investigating rendering using state-of-the-art programmable GPU cards through vertex and fragment programs. Complex 3D objects such as buildings and trees with large number of polygons, severely affect the rendering performance. Discrete multiresolution representation of these objects and their run-time selection may further increase the rendering speed.

REFERENCES

Agrawal, Anupam et al., 2004a. TREND: Adaptive Real-time View-dependent Level-of-detail-based Terrain Rendering. *Proceedings IT++: The Next Generation - the 39th Annual National Convention of Computer Society of India (CSI)* held in Mumbai, pp. 146-157.

Agrawal, Anupam et al., 2004b. Dynamic Multiresolution Level of Detail Mesh Simplification for Real-time Rendering of Large Digital Terrain Models. *Proceedings IEEE India Annual Conference 2004 (INDICON-2004)* at IIT, Kharagpur, pp. 278-282.

Agrawal, Anupam et al., 2005. An Approach to Improve Rendering Performance of Large Multiresolution Phototextured Terrain Models using Efficient Triangle Strip Generation. *Paper presented in IEEE IGARSS-2005* held in Seoul, Korea during July 25-29, 2005.

Cignoni, P. et al., 1998. A Comparison of Mesh Simplification Algorithms. *Computers and Graphics, 22(1)*, pp. 37-54.

De Boer, W. H., 2000. Fast Terrain Rendering Using Geometrical MipMapping. http://www.flipcode.com/articles/article_geomipmaps.pdf.

Dollner, J. et al., 2000. Texturing Techniques for Terrain Visualization. *In Proceedings of IEEE Visualization'2000*, pp. 207-234.

Douglass, D. et al., 1999. Real-Time Visualization of Scalably Large Collections of Heterogeneous Objects. *In Proceedings of IEEE Visualization'99*, pp. 437-440.

Duchaineau, M. et al., 1997. ROAMing Terrain: Real-time Optimally Adapting Meshes. *In Proceedings of IEEE Visualization'97*, pp. 81-88.

Lindstrom, P. et al., 1996. Real-Time Continuous Level of Detail Rendering of Height Fields. *In Proceedings of ACM SIGGRAPH'96*, pp. 109-118.

Lindstrom, P. and Pascucci, V., 2001. Visualization of Large Terrains made easy. *In Proceedings of IEEE Visualization'2001*, pp. 363-370.

Luebke, D., 2001. Survey of Polygon Simplification Algorithms. *IEEE Computer Graphics and Applications, 21(3)*, pp.24–35.

Luebke, D. et al., 2003. *Level of Detail for 3D Graphics*, Morgan Kaufmann Pub., pp. 19-46.

Pajarola, R., 1998. Large Scale Terrain Visualization using the Restricted Quadtree Triangulation. *In Proceedings of IEEE Visualization'98,* pp. 19-26.

Reddy, M. et al, 1999. TerraVision II: Visualizing Massive Terrain Databases in VRML. *IEEE Computer Graphics & Applications, Vol. 19(2)*, pp. 30-38.

Szenberg, F. et al., 1997. An Algorithm for the Visualization of a Terrain with Objects. *In X Brazilian Symposium on Computer Graphics and Image Processing*, pp. 103-110.

Vitter, J.S., 2001. External Memory Algorithms and Data Structures: Dealing with Massive Data. *ACM Computing Surveys, Vol. 33, Issue 2*, pp. 209-271.

Table 1:Performance Analysis of the Adaptive LOD Algorithm (without Objects).

Terrain Rendering (without objects)	**Avg. no. of Triangles**	**Avg. Frame per Second**
Full resolution (considering 9 tiles only)	1327104.00	01.63
View frustum culled surface	268338.82	05.96
Adaptive LOD algorithm (τ=4)		
1. Using triangle list	16764.25	58.63
2. Using triangle fan	16764.25	76.47
3. Using triangle strip (without using indexed vertex array)	19763.86	112.65
4. Using triangle strip (with using indexed vertex array)	19752.95	128.89

Table 2: Performance Analysis of the Adaptive LOD Algorithm (with Objects).

Terrain rendering (with Objects)	Avg. frames per second
1. Using total 5160 Objects (Opengl: 410, Billboards: 4750)	116.60
2. Using total 5170 Objects (Complex: 10, Opengl: 410, Billboards: 4750)	112.61

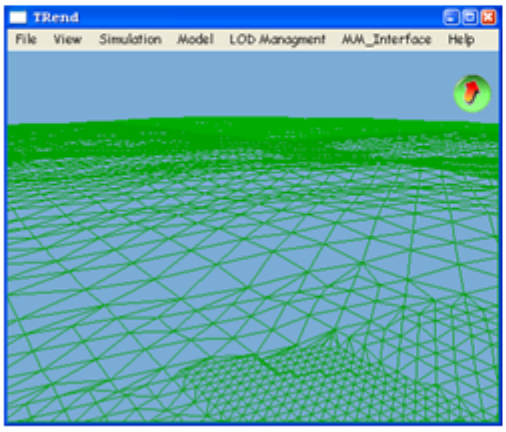

Figure 7: Level of detail wiremesh view (τ=4).

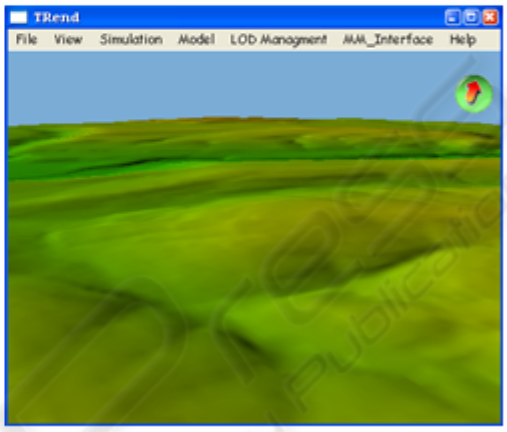

Figure 8: Corresponding Phototextured view.

(a) (b)

Figure 9: Display of point-location based objects.

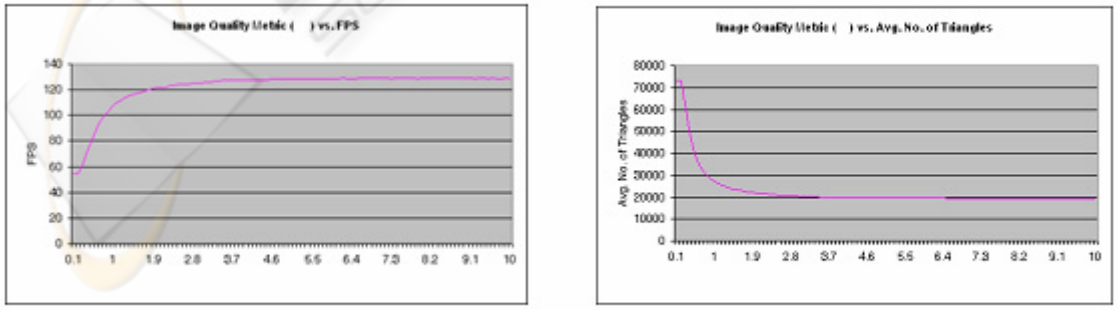

Figure 10: Effect of Image Quality Metric (τ) on Rendering Performance.