Interactive Rendering of Huge 3D Meshes in Cloud Computing

Daeyoung Kim and Haeyoung Lee
Computer Engineering Dept.
Hongik University
Seoul, Korea
dykim99@gmail.com, leeh@hongik.ac.kr

***Abstract* — This paper presents a new hierarchical representation of huge 3D meshes for fast and seamless rendering in cloud computing. Shape-outlines, simplified meshes, and uniform mesh partitions construct a hierarchy. Our hierarchy enables on-demand rendering of huge 3D meshes in cloud computing.**

Keywords-cloud computing, 3D meshes, interactive rendering, hiearchical representation.

I. INTRODUCTION

Rapid advances in 3D scanning technologies now enable us to create huge and exquisite 3D meshes for medical imaging and cultural heritage preservation. Nevertheless, it is hard and even impossible to render huge meshes on consumer computers and mobile devices due to limited resources. Various techniques such as mesh compression [1], [2], simplification [3], [4] and chartification [2] allow the transfer and display of huge meshes on mobile devices; however, interactive rendering in real time is hard to achieve using these methods. Though image-based rendering [5] has recently been introduced, pre-rendered images and grid-based sparse meshes cannot provide detailed views of original meshes. Moreover, these methods do not allow for the control of file size. Advances in CPU-related technologies have dramatically decreased CPU processing times so that I/O time contributes to almost the entire processing time. Uniformly sized files optimize I/O processing time and are especially necessary for mobile computing.

In this paper, we present an interactive rendering method of a hierarchical data structure for huge meshes for cloud computing platforms. Moreover, with our method the file size for each of a series of files for hierarchical data structures can be uniformly controlled for optimized and predictable I/O processing time.

The remainder of the paper is organized as follows: the basics of hierarchical rendering are described in detail in Section II; uniform mesh partitioning and simplifications are explained in Section II, parts A and B; our interactive view modes are listed in Section II, parts C through E; our conclusions and future work are presented in Section III.

II. HIERARCHICAL 3D MESH RENDERING

New hierarchical representations of huge 3D meshes allow for fast and seamless rendering of 3D meshes in cloud computing. The hierarchical display structure for a large 3D mesh is composed of several view options: a thumbnail-preview mode (TP), a coarse-whole-view mode (CWV), a zoomed-sector-view mode (ZSV), and finally a deep-zoom-of-the-mesh mode (DZM). Shape-outlines (TP mode), simplified meshes (CWV, ZSV modes), and the original mesh partitions (DZM mode) hierarchically represent large 3D meshes. For example, Table I depicts a huge mesh David with 28,184,526 vertices totaling 1.1 GB which cannot be loaded and displayed on a mobile device. Using our interactive rendering method, David can be displayed in real time on a mobile device with a hierarchical data structure as illustrated in Fig. 1. First, David is selected from a 3D shape-outline in TP mode in (a). A selected CWV is then generated with a simplified mesh of only 10,820 vertices in (b). After selecting the head in (b), a more detailed mesh of 10,649 vertices is rendered in ZSV mode as depicted in (c). For a DZM-mode view of the eye, partitions of the original mesh having 5,625 vertices are loaded and rendered in (d).

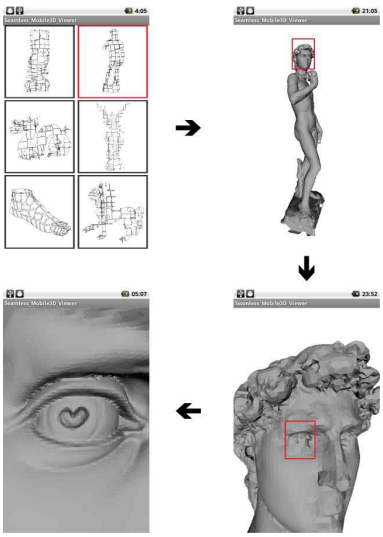


Figure 1. A hierarchical rendering of a huge 3D mesh on a mobile device[1]. David has 28,184,526 vertices at 1.1GB. (a) 3D shape-outlines in TP mode; (b) simplified David of 10,820 vertices; (c) more detailed head of 10,649 vertices; (d) original resolution eye of 5,625 vertices fully rendered.

An overview of our interactive 3D mesh rendering for cloud computing is depicted in Fig. 2. A huge mesh is

This work was supported by Seoul R&DB program (ST100035).
[1] A mobile device ODROID-7 with Samsung S5PC 110 Cortex-A8 1Ghz CPU and 512MB RAM.

 ISBN: 978-1-61208-187-8

TABLE I. LOADING AND RENDERING TIME ON A MOBILE DEVICE[2]

	Original Model		Our Simplified Model	
File Name	***Vertices***	***Time(s)***	***Vertices***	***Time(s)***
feline	49,864	3.48	10,379	0.59
foot	160,226	9.00	8,324	0.56
dragon	399,332	25.69	9,797	0.61
ihigenie	351,750	23.90	9,999	0.62
bddha	541,366	33.99	10,275	0.63
xyzrgb_dragon	3,609,455	N/A	9,589	0.60
lucy	14,027,872	N/A	10,180	0.64
david	28,184,526	N/A	10,820	0.66
Average		**19.21**		**0.61**

Uniformly partitioned and simplified meshes provide a hierarchical representation for huge 3D meshes so that interactive renderings on mobile devices can be performed with optimized and predictable processing times.

uniformly partitioned based on a user-specified equal number of vertices for uniform I/O processing time. Then a 3D shape-outline of each uniform partition is extracted and simplified through its own boundaries. Mesh simplifications in multi-resolution are then executed by calculating the representative vertex for a group of vertices in each partition. The number of partitions can be controlled by the user allowing the file size of a simplified mesh to be easily manipulated. This enables the client to transfer and render hierarchical structures of huge 3D meshes according to the user's interaction with the server in cloud computing.

Partitioned mesh files, simplified meshes, and a shape-outline are generated and stored on a server as a hierarchy automatically whenever a huge mesh is uploaded. Then a client can access the hierarchy starting from a shape-outline as shown in Fig.1. Our work will add interactive mesh simplifications to provide appropriate simplified meshes according to a user's choice of views for server-side processing in the future.

A. *Uniform Mesh Partitioning*

The main goal of partitioning a large mesh is to minimize processing time while maintaining load balance. The CPU in most systems today have improved radically resulting in input-output (I/O) processing becoming the main factor in the overall processing time. Uniform partitioning is the division of a large mesh into partitions with an equal number of user-specified vertices. Uniform partitioning is essential for a 3D mesh in cloud computing, so as to enable the assignment of standardized times to the processing of each partition as well as to optimize I/O processing time. Typically, mesh partitioning has been implemented by clustering vertices or faces. Clustering has been accomplished through either space subdivisions [1][7] or incremental additions [2][6]. The octree method provides fast hierarchical clustering [1][7]. However, the numbers of vertices or faces in partitions are varied because the division is performed not by the numbers of vertices or faces but by the sizes of the cells. K-means clustering [6] can generate partially uniform mesh partitions. However, it does not

[2] A smart phone LG-SU660 with 1GHz Dual Core CPU and 512MB RAM.

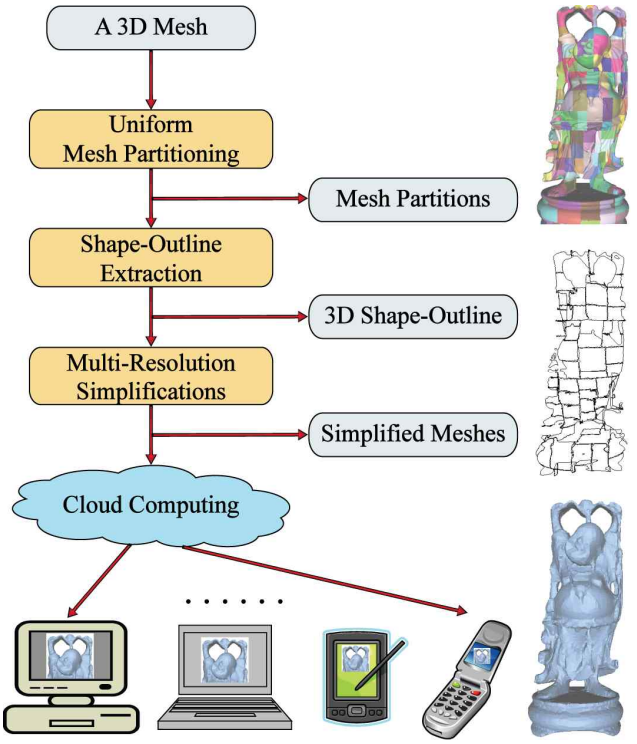


Figure 2. Overview of our interactive 3D mesh rendering in cloud computing.

provide uniform mesh partitioning and hierarchical clusters. Also, initial positions of random seeds must be carefully selected and an elaborate cost function must be designed to attain quality results. Optimization takes a great deal of time requiring many repetitions for large 3D meshes [2].

Our algorithm constructs a kd-tree for a mesh. Each cell in the kd-tree represents a vertex cluster which forms a single partition of the mesh. For a given mesh, our kd-tree divides space based on the object median where the objects are vertices of the mesh. Our kd-tree splits a cell into two sub-cells each containing half the vertices of the cell. Instead of cycling the axis from x to y to z-axis for a perpendicular splitting plane, our kd-tree determines the axis adaptively according to the longest axis of a bounding box. Compactness is a quantity for measuring the degree to which a shape is compact. Given a partition with area w and perimeter p, we define the compactness c of the partition as a ratio of its squared perimeter p2 to its areas w [9].

$$c = \frac{p^2}{4\pi w} \quad (1)$$

A square figure has better compactness than a long thin rectangular figure. To avoid long thin shaped partitions, our algorithm considers compactness when determining an axis for perpendicular splitting planes to subdivide cells in the kd-tree. Fig. 3 depicts the steps from level 1 to level 4 of the kd-tree in a simplified 2D format. The dotted-line is the bounding box of vertices in a cell. The solid lines are

 ISBN: 978-1-61208-187-8

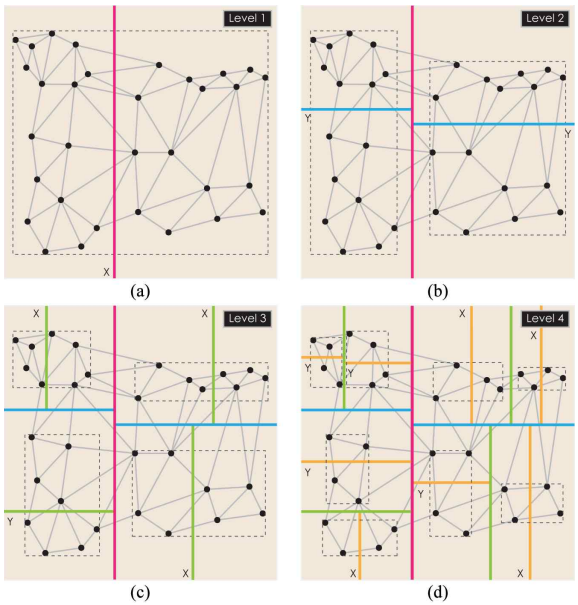


Figure 3. An example of our kd-tree construction in 2D. Our kd-tree is based on the uniform number of vertices in each cell. An axis to be split is adaptively determined to lower the compactness of the cell.

determined by, and are perpendicular to, the longer of the two axes, x and y, of the bounding box and colored red for two cells in level 1, blue for four cells in level 2, green for eight cells in level 3, and yellow for sixteen cells in level 4. The axes for cells at the same level may be chosen differently depending on the shape of the bounding boxes as depicted in Fig. 3. Median values are computed to split vertices in half. Finally, sixteen uniform partitions of the mesh are created from sixteen clusters of vertices in sixteen leaf-cells in the kd-tree. To construct a kd-tree of a mesh, a median value of the vertices in a cell needs to be determined so as to split a cell into two subcells with equal numbers of vertices. For an out-of-core mesh which has more data than the size of the main memory, external sorting needs to be applied; however, external sorting takes a lot of time. Therefore, we plan to introduce an improved out-of-core sorting method to find median values.

In Fig. 4, two previous partitioning methods are compared with our method for a model foot of 40,058 vertices in (a). Partitioning results are listed in (b) by k-means clustering, (c) by octree-based clustering, and (d) by our kd-tree based clustering. The numbers of vertices in each partition are charted in (e). K-means clustering generates 128 partitions in 6.22 seconds with a compactness measure of 3.155. Octree clustering runs fast in 3.76 seconds with a compactness of 1.806 for 126 partitions. Our kd-tree clustering generates 128 uniform partitions in 3.85 seconds with a compactness of 1.929. Only our kd-tree based clustering creates uniform partitioning with quality shapes in a relatively fast processing time.

B. *Mesh Simplification Using Our Mesh Partitioning*

A uniform number of vertices in a partition plays a key

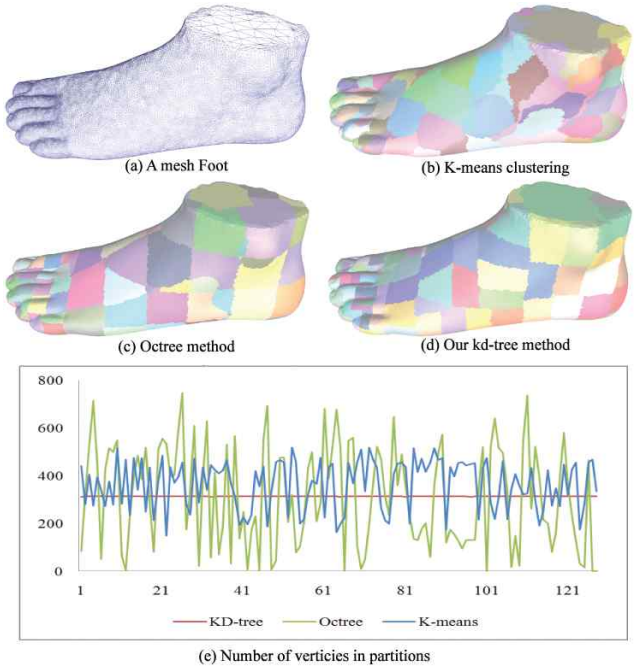


Figure 4. Examples of mesh partitioning. A mesh Foot of 40,058 vertices and 80,112 faces is rendered in wire frame in (a). Partitioning results are listed in (b) by k-means clustering, (c) by octree-based clustering, and (d) by our kd-tree based clustering. The numbers of vertices in each partition are charted in (e).

role in the quality of the mesh simplification. A single representative vertex for a partition was calculated for the vertices of the partition. Triangulations were performed with simplified vertices according to the original connectivity. The size and the shape of simplified triangle faces are more regular with our kd-tree method since each simplified vertex represents a uniform number of vertices, whereas each simplified vertex using the octree method represents various numbers of vertices as depicted in Fig. 5. In (a), the mesh is simplified to 1,104 vertices using our kd-tree method whereas in (b) the mesh is simplified to 1,206 vertices using the octree method. The mean distortion to the original mesh is 0.4149 by our kd-tree and 0.4563 with the octree [8]. Our simplification preserves the original shape better with better triangulation.

C. *3D Shape-Outlines: Interactive 3D Previews*

For 2D images, TP mode provides small thumbnail images so that a user can easily and quickly identify and select a specific image. Until present, an interactive TP mode for 3D meshes has not been available. As such, we offer our TP mode which uses shape-outlines for interactive 3D mesh previews to dramatically reduce file size. As shown in Fig. 1(a), a series of shape-outlines can easily be displayed on mobile devices with no need for file names. A shape-outline also depicts how the mesh is partitioned. As illustrated in Fig. 6, each TP shape-outline can be interacted with to translate or rotate the thumbnail with no need to fully display the mesh.

 ISBN: 978-1-61208-187-8

Figure 5. Examples of simplified meshes with a mesh Foot. In (a), the mesh is simplified to 1,104 vertices using our kd-tree method whereas in (b), the mesh is simplified to 1,206 vertices using the octree method.

D. *Simplified Meshes in Multi-Resolution*

Simplified meshes in multi-resolution can provide CWV and ZSV modes of 3D meshes. As shown in Fig. 7, the more detailed Buddha of 9,539 vertices in (b) provides higher resolution than simply an enlarged but degraded view of the simplified mesh of 2,569 vertices in (a).

E. *Mesh Partitions for the Closest View*

Finally, for DZM-mode views of meshes, uniformly partitioned files of the selected area of the original mesh are transferred and rendered as shown in Fig. 1(d). The number of vertices in each partition can be specified by a user to provide optimized and predictable processing time for each partition.

III. CONCLUSION AND FURTURE WORK

This paper introduced a hierarchical representation of 3D meshes for interactive rendering in cloud computing. As listed in Table I, the rendering of 3D meshes on a mobile device took 19.21 seconds on average while huge meshes could not be loaded due to device memory limitations. With our hierarchical method, interactive rendering can be provided in real time in about 0.6 seconds on average even for huge meshes. In our future work, we will investigate how to automatically control simplification levels on the server or the client. Our research has led us to conclude that texture mapping to simplified meshes should be further studied. Moreover, how to approximate texture coordinates for simplified meshes also needs further investigation.

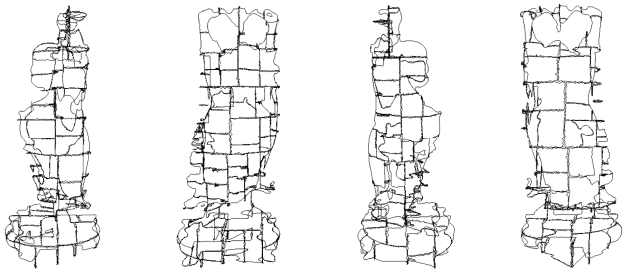
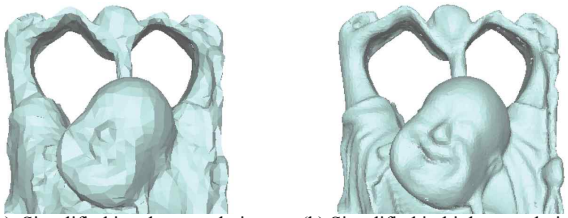


Figure 6. A shape-outline of a model Buddha in various views. A user can interactively control TP mode views of the shape-outline.



(a) Simplified in a low resolution (b) Simplified in higher resolution

Figure 7. Simplified meshes in multi-resolution for a mesh Buddha. Rather than zooming in to a lower resolution of the simplified mesh in CWV mode in (a), rendering can automatically switch to ZSV mode to get a higher resolution zoom.

REFERENCES

[1] D. Kim, S. Lee, H. Lee, and S. Cho, “A distance-based compression of 3D meshes for mobile devices,” IEEE Trans. Consumer Electron., vol. 54, no. 3, pp. 1398-1405, 2008.

[2] S. Choe, J. Kim, H. Lee, and S. Lee, “Random accessible mesh compression using mesh chartification,” IEEE Trans. Visualization and Computer Graphics, vol. 15, no. 1, pp. 160-173, 2009.

[3] P. Cignoni, C. Montani, C. Rocchini, and R. Scopigno, “External memory management and simplification of huge meshes,” IEEE Trans. Visualization and Computer Graphics, vol. 9, no. 4, pp. 525-537, 2003.

[4] S. Schaefer and J. Warren, “Adaptive vertex clustering using octrees,” SIAM Geometric Design and Computing, 2003.

[5] Y. Okamoto, T. Oishi, and K. Ikeuchi, ”Image-Based Network Rendering of Large Meshes for Cloud Computing,” International Journal of Computer Vision, vol. 94, no. 1, pp. 23-35, August 2011.

[6] S. Lloyd, “Least square quantization in PCM,” Information Theory, IEEE Transactions, vol. 28, no. 2, pp. 129-137, 1982.

[7] P. Cignoni, C. Montani, C. Rocchini, and R. Scopigno, “External memory management and simplification of huge meshes,” IEEE Trans. Visualization and Computer Graphics, vol. 9, no. 4, pp. 525-537, 2003.

[8] P. Cignoni, C. Rocchini and R. Scopigno, “Metro: measuring error on simplified surfaces,” Computer Graphics Forum, Blackwell Publishers, vol. 17, no. 2, pp. 167-174, June 1998.

[9] M. Garland, A. Willmott, and P. Heckbert, “Hierarchical face clustering on polygonal surfaces,” Proc. ACM Symposium on Interactive 3D Graphics, pp. 49-58, 2001.

 ISBN: 978-1-61208-187-8