View-dependent Transmission of Three-dimensional Mesh Models Using Hierarchical Partitioning

Sung-Yeol Kim*[a], Jeong-Hwan Ahn and Yo-Sung Ho

Kwangju Institute of Science and Technology (K-JIST)
1 Oryong-dong Puk-Gu, 500-712, Kwangju, KOREA

ABSTRACT

In this paper, we propose a new scheme for sending three-dimensional (3-D) mesh models. In order to make a view-dependent representation of 3-D mesh models, we combine sequential and progressive mesh transmission techniques. After we partition a 3-D mesh model into a hierarchical tree, we determine the amount of information for each submesh. Then, we can send the 3-D model information by view-dependent selection with mesh merging and splitting operations. Experimental results have demonstrated that the proposed scheme can send mesh information adaptively through mesh merging and splitting operations and provides good visual quality in a limited bandwidth channel.

Keywords: View-dependent transmission, Hierarchical partitioning, Mesh merging and splitting.

1. INTRODUCTION

As demands for high-quality visual services are increasing, three-dimensional (3-D) mesh models are widely used in various multimedia applications, such as Internet games, movies, and educational tools. The 3-D models are often represented with triangular meshes and they are defined by geometry, connectivity, and photometry information. Because of the tremendous amount of those data, it is hard to handle the 3-D models for storage, rendering, and transmission. Especially, research on efficient 3-D mesh coding and transmission is an important part in web-based applications; however, relatively little work has been done on this topic until now.

When we send 3-D mesh models, progressive transmission is widely used since they can be rendered just after reception of mesh information with no waiting time. In the progressive mesh, whose concept was first proposed by Hoppe in 1996 [1], we convert a 3-D mesh model into a continuous resolution form. The progressive mesh is used in smoothing level-of-detail approximations and mesh compression as well as progressive transmission. Hoppe also proposed view-dependent refinement of progressive meshes based on selective refinement [2].

In sequential mesh transmission, which is another way to send 3-D mesh models, we divide a 3-D mesh model into several independent pieces by a mesh partitioning algorithm, such as a multi-seed vertex traversal technique [3]. Each partiton, which is called a submesh, is then transmitted sequentially. If we send the 3-D mesh model sequentially, spatial transmission errors may affect only some limited parts of the model, not the entire model, at the receiver side. Recently, Yang [4] has proposed a view-dependent progressive coding scheme based on mesh partitioning.

In this paper, we propose a new scheme to send 3-D mesh models. We combine sequential and progressive mesh transmission techniques to make a view-dependent representation of 3-D mesh models. By transmitting visible parts before invisible parts, we can provide good visual quality in a bandwidth-limited channel and also reduce the initial waiting time in various web-based applications. After we convert the 3-D mesh model into a new representation, called a hierarchical mesh, and store it in the server, we determine the amount of transmitted information for each submesh, called a resolution. We can then send the 3-D mesh model by view-dependent selection with mesh merging and splitting operations according to the resolution.

This paper is organized as follows. Sec. 2 explains hierarchical mesh partitioning, and Sec. 3 describes the proposed view-dependent transmission scheme. After we provide experimental results in Sec. 4, we make conclusions in Sec. 5.

* sykim75@kjist.ac.kr; phone: +82-62-970-2258; fax: +82-62-970-2204; http://visual.kjist.ac.kr/~sykim; 1 Oryong-dong, Buk-Gu, 500-712, Kwangju, Korea.

Visual Communications and Image Processing 2003, Touradj Ebrahimi, Thomas Sikora, Editors, Proceedings of SPIE Vol. 5150 (2003) © 2003 SPIE · 0277-786X/03/$15.00

2. HIERARCHICAL MESH PARTITIONING

Mesh partitioning techniques are used to divide a given 3-D mesh model into several independent pieces. We call the independent piece as a submesh. When submeshes are transmitted instead of the entire model, it is useful for storage and transmission since spatial transmission errors will not affect the entire model at the receiver side. In this paper, we propose a new mesh structure to represent 3-D mesh models. We call this new structure as a hierarchical tree and the partitioning process as a hierarchical partitioning. We can generate a hierarchical tree for a 3-D mesh model when we specify the number of the initial submeshes and the number of levels.

The input data for hierarchical partitioning is geometry and connectivity information of the 3-D mesh model. We select the initial vertices as many as the number of the initial submeshes. For mesh partitioning [6], we can apply a multi-seed traversal technique [3] that is one of the well-known partitioning techniques. We perform hierarchical partitioning recursively until we have the specified number of levels. In other words, one parent submesh is divided into child submeshes as many as the number of the initial submeshes in the current level. Then, if the current level is less than the specified one, all child submeshes become parent submeshes in the next level and each submesh is partitioned again. Consequently, we construct the parent-child relationship in a tree structure. Fig. 1 shows an example of hierarchical partitioning when the number of the initial submeshes is two and the specified number of levels is three.

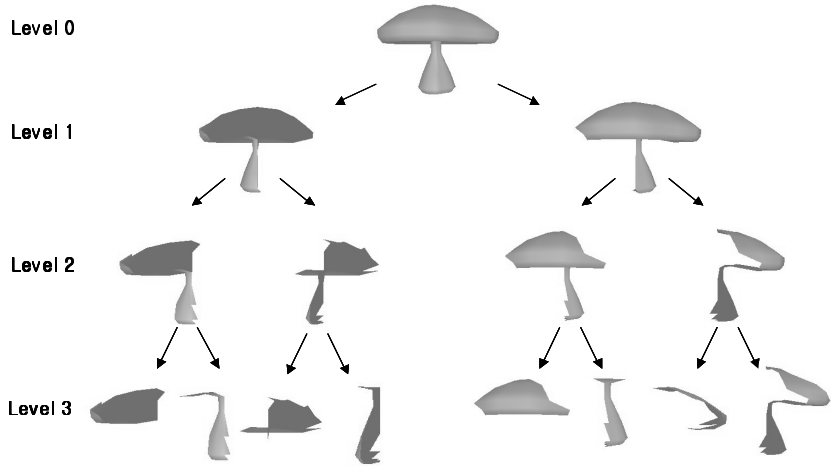


Figure 1. Hierarchical partitioning

2.1 Initial vertex selection

In order to begin the hierarchical partitioning operations, we should select the initial vertices as many as the number of the initial submeshes. The selection of the initial vertices is important since shapes of submeshes are determined by their positions. In this paper, we use the K-means clustering algorithm [7] to select initial vertices. First, we set the number of the initial submeshes K that is the total number of child submeshes obtained from a parent submesh. The K-means algorithm performs to find positions of initial vertices as many as K. The initial vertices are selected by.

1. Choose K initial center vertices $z_1(1)$, $z_2(1)$,..., $z_K(1)$ by selecting arbitrary K vertices in the parent submesh.

2. At the k-th iterative step, distribute the input vertices {X} among the K cluster domains by

$$x \in S_j(k) \quad if \quad \| x - z_j(k) \| < \| x - z_i(k) \| \tag{1}$$

for all i = 1, 2,...,K, i ≠ j , where $S_j(k)$ denotes the set of vertices whose center is $z_j(k)$.

3. From Step 2, compute new cluster centers $z_j(k+1)$, j = 1,2,...*K*, such that the sum of squared distances from all points in $S_j(k)$ to the new cluster center is minimized. In other words, the new cluster center $z_j(k+1)$ is computed to minimize the performance index

$$J_j = \sum_{x \in S_j(k)} \| x - x_j(k+1) \|^2, \quad j = 1, 2, \ldots, K \tag{2}$$

Here, we note that $z_j(k+1)$ is simply the sample mean of $S_j(k)$. Therefore, the new vertices center is given by

$$z_j(k+1) = \frac{1}{N_j} \sum_{x \in S_j(k)} x \qquad \text{j} = 1, 2, \ldots, K \tag{3}$$

where N_j is the number of samples in $S_j(k)$.

4. If $z_j(k+1) = z_j(k)$ for j = 1,2,...,K, the algorithm has converged and the procedure will be terminated. Otherwise, go to Step 2. If the procedure is terminated, the new center vertices $z_1(k+1)$, $z_2(k+1)$,..., $z_K(k+1)$ become initial vertices.

The behavior of the K-means clustering algorithm is influenced by the number of cluster centers specified. Therefore, the mesh partitioning from these initial vertices can be operated in an optimal manner. On the contrary, the K-means clustering algorithm needs more computational time than other algorithms, such as a maximum-distance algorithm. However, we do not need the real-time partitioning. Fig. 2(a) and Fig. 2(b) show selections of initial vertices by the maximum-distance algorithm [4] and the K-means algorithm, respectively, when the number *K* of the initial submeshes is three. From Fig. 2, we can notice that the initial vertex positions selected by the K-means algorithm are more optimal than those selected by the maximum distance algorithm.

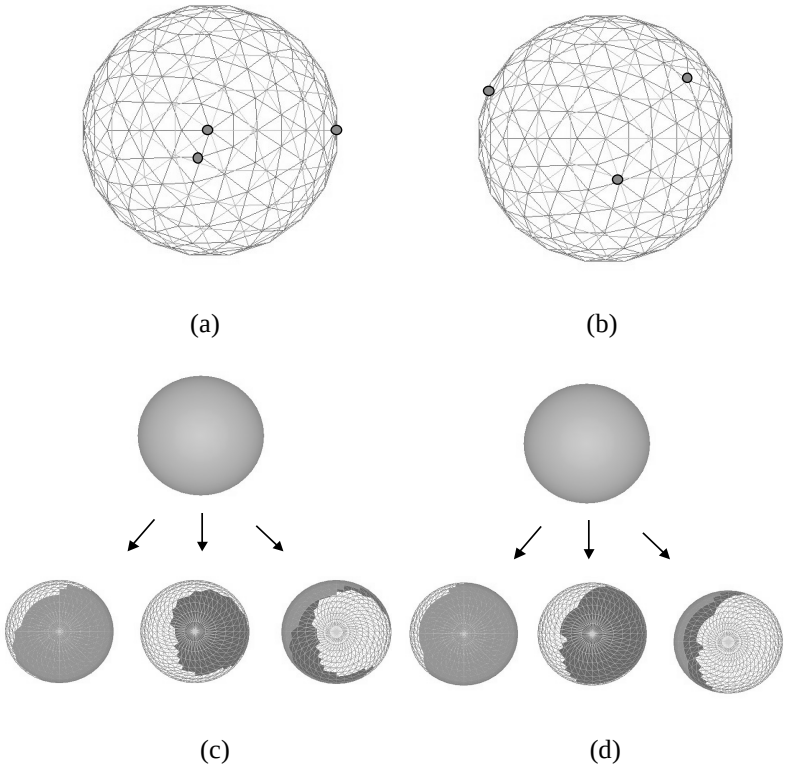


Figure 2. Initial vertex selection: (a) maximum distance algorithm, (b) K-means algorithm, (c) the result of (a), (d) the result of (b)

2.2 Hierarchical mesh

Each submesh in the last level of the hierarchical tree is represented by a progressive mesh, which consists of a base layer and several enhancement layers. The progressive mesh is usually obtained by the inverse operation of mesh simplification [8,9,10]. The base layer is a simplified version of the mesh model and has a minimum data regardless of view-dependency. On the other hand, enhancement layers contain additional resolution information, which is related to the viewing condition, and they can be removed during mesh simplification. In the base layer, the boundary vertices are always connected to the initial vertex [4]. In this paper, the multi-layer representation of the 3-D mesh model is constructed based on the hierarchical tree. The 3-D mesh model can be represented by a hierarchical mesh. Fig. 3 shows an example of the hierarchical mesh.

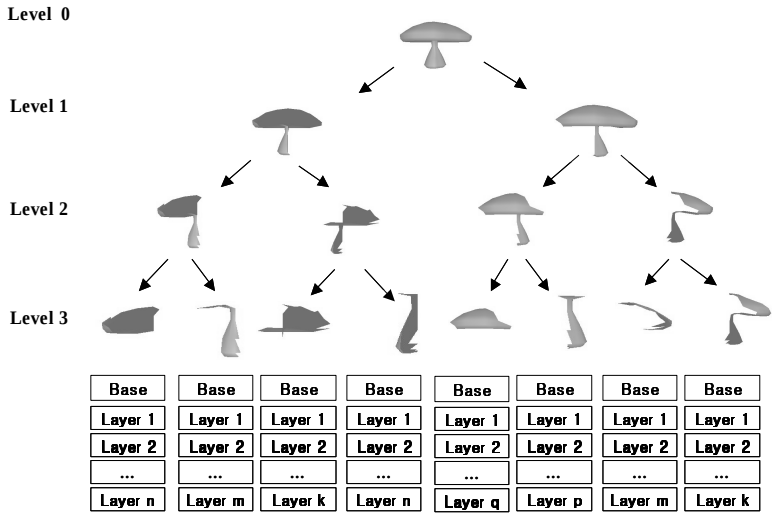


Figure 3. Hierarchical mesh

When the i-th submesh in the last level of the hierarchical tree is denoted by M_i, we can get the base layer M^0_i by edge contraction. The removed information by edge contraction is used in the enhancement layers. Eqs. 4 shows the procedure for making the base layer of the i-th submesh.

$$(M_i = M_i^k)\ \overrightarrow{econ_i^{k-1}}\ \ldots\ \overrightarrow{econ_i^1}\ M_i^1\ \overrightarrow{econ_i^0}\ M_i^0 \tag{4}$$

Eqs. 5 represents the reconstructed procedure. After *k* times of the vertex splitting operation, we have

$$M_i^0\ \overrightarrow{vsplit_i^0}\ M_i^1\ \overrightarrow{vsplit_i^1}\ \ldots\ \overrightarrow{vsplit_i^{k-1}}\ (M_i = M_i^k) \tag{5}$$

3. VIEW-DEPENDENT TRANSMISSION OF 3-D MESH MODEL

After a 3-D mesh model is converted into a hierarchical mesh and stored in the server, the server should set the initial viewing parameter for the mesh model before transmission. In order to send the stored model, we first determine the amount of transmitted data, which is called a resolution. After deciding resolutions, we send the information of the 3-D mesh model by view-dependent selection with mesh merging and splitting operations.

3.1 Resolution decision

Before sending a 3-D mesh model to the receiver, we first decide the resolution *R*, which indicates the degree of visibility of its submeshes. In other words, the resolution *R* is related to the amount of the transmission data. The

resolution is calculated by checking the intersection angle Θ between the normal vector n_j of the j-th vertex in a submesh and the viewing parameter V of the viewer. If the intersection angle Θ is larger than 90^0, the j-th vertex is invisible. Otherwise, the j-th vertex is visible. Θ can be calculated by

$$\theta = \arccos\left(\frac{\vec{V} \cdot n_j}{|\vec{V}||n_j|}\right) \tag{6}$$

There are two situations when we decide the resolution of submeshes: static viewing and dynamic viewing. In static viewing, we can determine resolutions with the initial viewing parameter. On the other hand, we re-estimate resolutions with the changed viewing parameter in dynamic viewing. Fig. 4 illustrates the resolution decision.

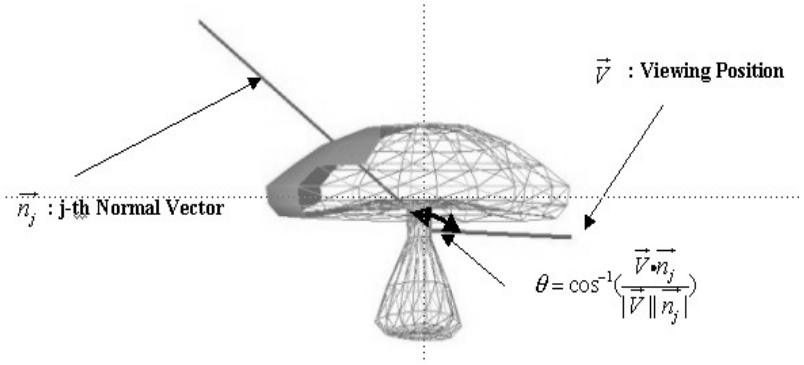


Figure 4. Resolution decision

When there is a submesh A, the representative angle Θ_A is the minimum angle among angles between all the normal vectors of A and the viewing parameter.

$$\theta_A = \min\left[\arccos\left(\frac{\vec{V} \cdot n_j}{|\vec{V}||n_j|}\right)\right] \tag{7}$$

The resolution R_A can be calculated by Eqs. 8. R_{max} denotes the total number of the enhancement layers of A

$$R_A = R_{\max} \times \cos\theta_A \tag{8}$$

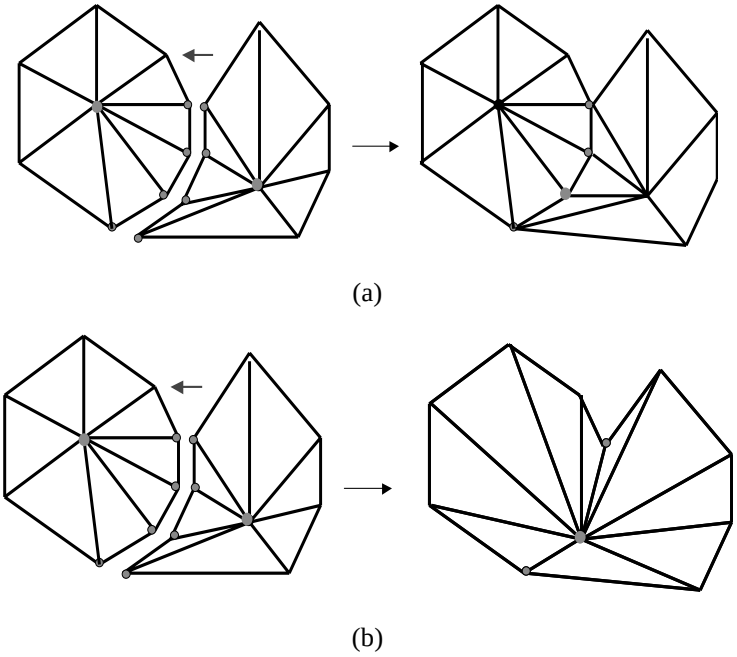
3.2 Mesh merging

Mesh merging gathers child submeshes $\{A_1, A_2, \ldots, A_K\}$ into their one-level upper parent submesh *A* according to their resolutions. Since submeshes act independently, they have joint boundary information coded redundantly. The smaller the size of the submesh is, the more joint boundary information is transmitted redundantly. When we set the number of vertex in the submesh to 32, the boundary information occupies about from 15% to 30% of the original mesh model information. However, if you send mesh models without partitioning, the mesh information can be distorted by transmission errors. Therefore, we need to select submeshes appropriately.

By mesh merging operations, we can reduce the joint boundary information and also select transmitted submeshes adaptively. After mesh merging, the resolution of the parent submesh is different locally since the server maintains resolutions of its child submeshes. Therefore, we can reduce the joint boundaries while maintaining the resolutions of its child submeshes. Moreover, the merged parent submesh can be a view-dependent submesh conceptually that is suitable for view-dependent transmission. Mesh merging is performed in the base layers of child submeshes.

There are two operations in mesh merging: visible mesh merging and invisible mesh merging. Visible mesh merging is performed among visible submeshes by connecting the common boundary information. On the other hand, invisible

mesh merging is performed among invisible submeshes. After invisible mesh merging, several base layers of child submeshes are changed into a base layer of the parent submesh. Fig. 5 illustrates visible mesh merging and invisible mesh merging operations.



(a)

(b)

Figure 5. Mesh merging: (a) visible mesh merging, (b) invisible mesh merging

3.3 Mesh splitting

Mesh splitting divides a parent submesh *A* into one-level lower child submeshes $\{A_1, A_2,\ldots, A_K\}$ according to their resolutions. This operation is the inverse of mesh merging. However, mesh splitting is performed conceptually not physically. That is, a mesh is not divided into several independent pieces as in partitioning. Mesh splitting is just for determining the amount of additional transmission data to the receiver. As we mention in Sec. 3.1, we decide resolutions of submeshes when the viewing parameter is changed. If the current resolution is higher than the previous resolution, we should send additional data of the difference between current and previous resolutions. On the other hand, if the current resolution is equal or lower than the previous resolution, we do not need to transmit additional data since enough mesh information was already transmitted.

There are two methods in mesh splitting: visible mesh splitting and invisible mesh splitting. In visible mesh splitting, a parent submesh is divided into several child submeshes conceptually. On the other hand, invisible mesh splitting separates a base layer into several base layers. Fig. 6 demonstrates invisible mesh splitting.

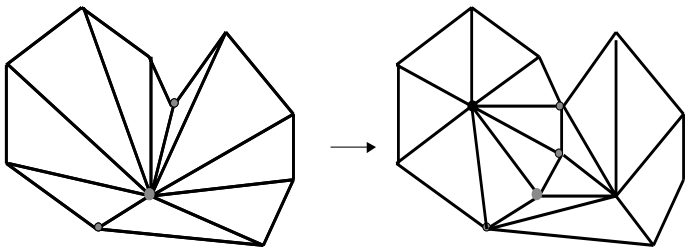


Figure 6. Invisible mesh splitting

3.4 View-dependent selection

The main idea of view-dependent selection is that submeshes are merged or split according to their resolutions before transmission. In this section, we consider two different situations: static viewing and dynamic viewing. The initial viewing parameter is used in static viewing, while the changed viewing position of viewer is used in dynamic viewing.

3.4.1 Static viewing

In static viewing, we assume that the viewing parameter at the receiver is not changed. In other words, there are no rotation, zooming, and transition operations. Generally speaking, the static viewing mode is used in the initial transmission of the 3-D mesh model. In static viewing, we perform the mesh merging operation.

Fig. 7 shows an example of the static viewing mode. A 3-D mesh model is represented by a hierarchical mesh. The sequential numbers indicate submeshes at each level. Submeshes 1, 4, 5, 6, and 7 at level 3 are visible, but submeshes 0, 2, and 3 are invisible. In static viewing, only submeshes with large circle are transmitted. Thin circles indicate visible submeshes and thick circles indicate invisible submeshes. As shown in Fig. 7, we perform visible mesh merging for submeshes 4, 5, 6, and 7 submeshes. In addition, invisible mesh merging is performed for submeshes 2 and 3. Therefore, we send a base layer of submesh 0 at level 3, a base layer and enhancement layers of submesh 1 at level 3, a base layer of submesh 1 at level 2, and a base layer and enhancement layers of submesh 1 at level 1.

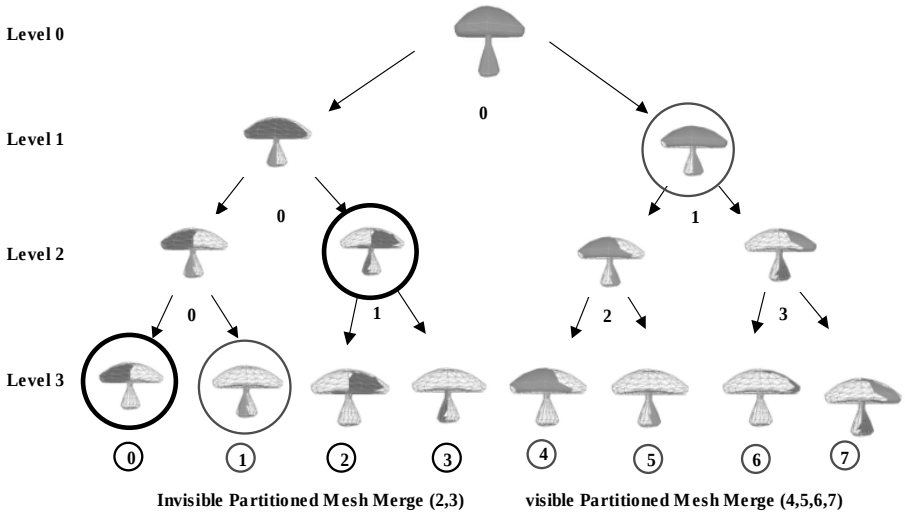


Figure 7. Static viewing

3.4.2 Dynamic viewing

In dynamic viewing, we assume that the viewing parameter is changed. In other words, there are rotation, zooming, and transition operations. The changed viewing parameter is sent to the server. When we transmit a 3-D mesh model in the static viewing mode, the invisible parts are distorted heavily. Therefore, when the viewer at the receiver side wants to explore the invisible parts, we should improve visual quality by additional information. In dynamic viewing, we perform the mesh splitting operation.

Fig. 8 shows an example of the dynamic viewing mode. When the viewing parameter is changed, we determine resolutions with the changed parameter. As a result, submeshes 0, 2, 4, 6, and 7 at level 3 become visible, but submeshes 1, 3, and 5 become invisible. After the mesh splitting operation, submesh 1 at level 1 is split into submesh 3 at level 2, submesh 4 at level 3, and submesh 5 at level 3. In addition, submesh 1 at level 2 is split into submeshes 2 and 3 at level 3. After deciding whether we should send additional data for the split submeshes, we retransmit the additional mesh information according to their resolutions.

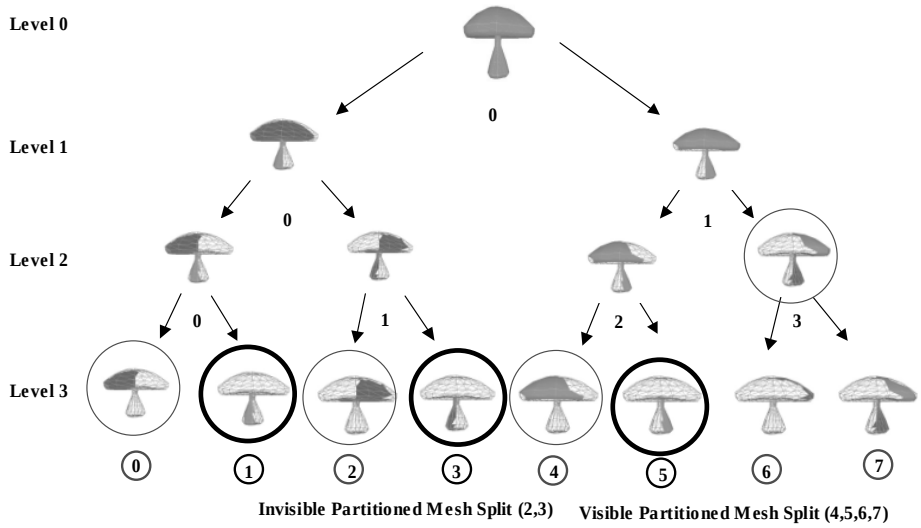


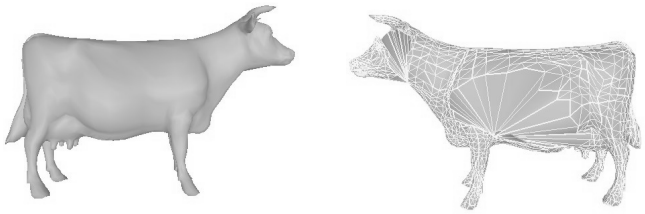
Figure 8. Dynamic viewing mode

4. EXPERIMENTAL RESULTS AND ANALYSIS

We have evaluated the proposed algorithm with several 3-D mesh models. All tested models have manifold topology. We show experimental results in both static and dynamic viewing conditions. We have also compared results of simple partitioning and hierarchical partitioning. The tested models include COW and MUSHROOM models. The COW model consists of 2903 vertices and 5804 faces. In order to represent the COW model by a hierarchical mesh, we have specified that the number of the initial submeshes is 2 and the number of levels is 5. The MUSHROOM model contains 226 vertices and 448 faces, and we have set the number of the initial submeshes to be 2 and the number of levels to be 3 for a hierarchical mesh.

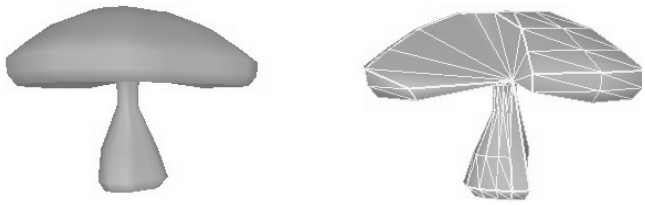
4.1. Static viewing

We have tested the COW and MUSHROOM models in the static viewing mode. The initial viewing parameter is (x = 0.0, y = 0.0, z = 1.0). Fig. 9 and Fig. 10 show the results of the COW and MUSHROOM models in static viewing. While Fig. 9(a) and Fig. 10(a) indicate the results of visible parts, Fig. 9(b) and Fig. 10(b) indicate the results of invisible parts, respectively. From Fig. 9 and Fig. 10, we notice that the visible parts of the tested models have no distortion since enough mesh information has been transmitted. However, the invisible parts have distortions since we have only sent the base layer. We could reduce the transmitted mesh information about 23% ~ 25% in the static viewing mode, as shown in Table. 1



(a) (b)

Figure 9. Result of COW in static viewing: (a) visible parts, (b) invisible parts



(a) (b)

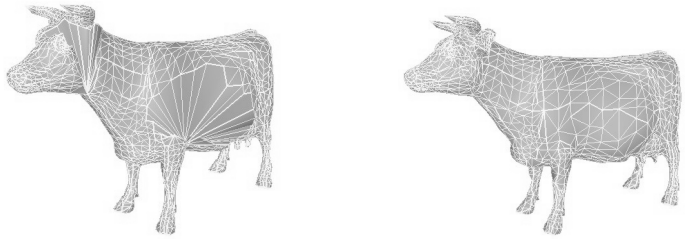
Figure 10. Result of MUSHROOM in static viewing: (a) visible parts, (b) invisible part

Table 1. Result in static viewing and dynamic viewing

Mesh model	Original mesh		Static viewing		Gain		Dynamic viewing		Addition	
	F1	V1	F2	V2	F2/F1	V2/V1	F3	V3	F3-F2	V3-V2
MUSHROOM	488	226	372	172	0.762	0.763	414	193	42	21
COW	5804	2903	4362	2189	0.752	0.755	4898	2457	536	268

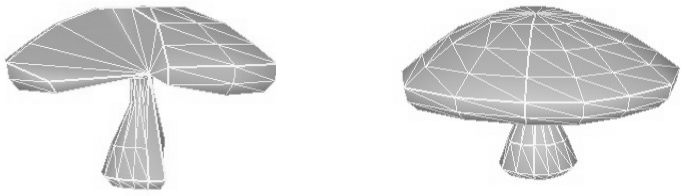
4.2. Dynamic viewing

We have changed the viewing parameter from (x = 0.0, y = 0.0, z = 1.0) to (x = 1.0, y = 0.0, z = -1.0). Fig. 11 and Fig. 12 show the results of the COW and MUSHROOM models in the dynamic viewing mode, respectively. From Fig. 11 and Fig. 12, we notice that the new visible parts have no distortion since we have sent additional mesh information to refine these parts. The additional data has 536 faces and 268 vertices in the COW model, and 42 faces and 21 vertices in the MUSHROOM model, respectively.



(a) (b)

Figure 11. Result of COW in dynamic viewing: (a) before, (b) after

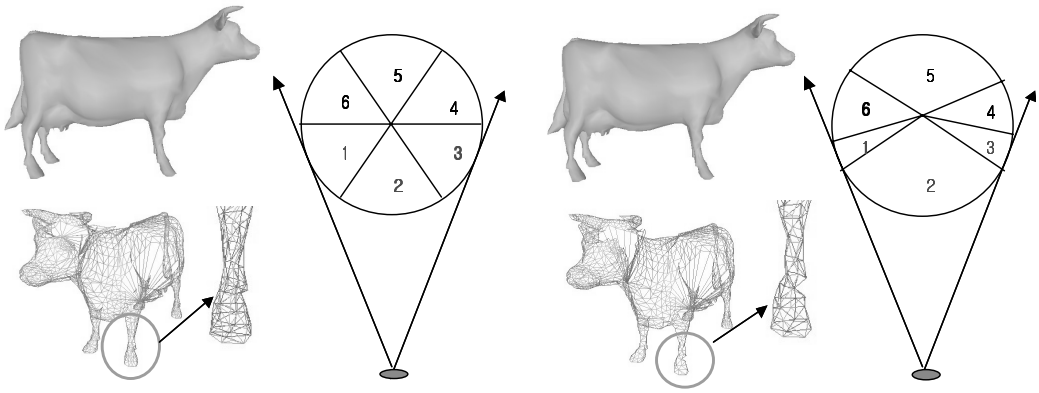


(a) (b)

Figure 12. Result of MUSHROOM in dynamic viewing: (a) before, (b) after

4.3. Hierarchical partitioning

We have compared the proposed hierarchical partitioning to a simple partitioning in the static viewing mode. In this experiment, we assume that both schemes have the same joint boundary information. Fig. 13 shows experimental results of the two schemes.



(a) (b)

Figure 13.Comparision: (a) simple partitioning, (b) hierarchical partitioning

From Fig. 13, we can observe that the visible parts are not distorted in both schemes. However, we can note that some unnecessary information is transmitted in the simple partitioning scheme in Fig. 13. When we consider that the goal of the static viewing mode is reduce the waiting time in web-based applications, the proposed scheme is more suitable. As shown in Table 2, we can reduce 8% ~ 10% of the transmitted mesh information than the simple partitioning scheme.

Table 2 Comparison of hierarchical partitioning and simple partitioning

Mesh model	Number of partitions	Original mesh		Simple partitioning		Hierarchical partitioning		Comparison			
		F1	V1	F2	V2	F3	V3	F2/F1	V2/V1	F3/F1	V3/V1
COW	18	5804	2903	4960	2429	4362	2189	0.855	0.836	0.752	0.755

5. CONCLUSIONS

In this paper, we have proposed a new scheme for sending 3-D mesh models. We have combined progressive and sequential transmission techniques to use view-dependency of 3-D mesh models. In order to send the 3-D mesh model efficiently, we represent the 3-D mesh model into a hierarchical mesh by hierarchical partitioning. Then, we send the 3-D mesh model with view-dependent selection by mesh merging and splitting operations. Although it takes a long time to send a 3-D mesh model because of the tremendous amount of data, we can reduce the transmitted data by focusing on the visible parts at the receiver. In addition, we can obtain improved visual quality in a bandwidth limited channel through view-dependent selection.

ACKNOWLEDGEMENT

This work was supported in part by Kwangju Institute of Science and Technology (K-JIST), in part by the Korea Science and Engineering Foundation (KOSEF) through the Ultra-Fast Fiber-Optic Networks (UFON) Research Center at K-JIST, and in part by the Ministry of Education (MOE) through the Brain Korea 21 (BK21) project.

REFERENCES

1. H. Hoppe, "Progressive Meshes," *Proc. SIGGRAPH-96*, pp.99-108, Aug. 1996.
2. H. Hoppe, "View-dependent Refinement of Progressive Meshes," *Proc. SIGGRAPH-97*, pp. 189-198, 1997
3. Z. Yan, S. Kumar and C-C.J. Kuo, "Error-resilient Coding of 3-D Graphics Models via Adaptive Mesh Segmentation," *Proc. IEEE Trans. Circuits and Systems for Video Technology*, vol. 11. no 7, July 2001.
4. S. Yang, C.S. Kim and C-C.J. Kuo, "View-dependent Progressive Mesh Coding Based on Partitioning," *SPIE VCIP-2002*, vol. 4671, pp 268-279, Jan. 2002.
5. J.C. Xia and A. Varshney, "Dynamic View-dependent Simplification for Polygonal Environments," *Proc. Visualization 96*, pp.327-334, 1996
6. Z. Yan, S. Kumar, J. Li and C-C.J. Kuo, "Robust Coding of 3D Graphics Models Using Mesh Segmentation and Data Partitioning," *Proc. IEEE int. Conf. Image Processing*, Oct. 1999
7. J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles*, Addison-Wesley Publishing Company, 1974.
8. A. Maria-Elena and S. Francis, "Mesh Simplification," *Eurographics 96*, pp.77-86, 1996
9. C. Gotsman, S. Gumhold and L. Kobblelt, "Simplification and Compression of 3D Meshes," *Tutorials on Multiresolution in Geometric Modeling*, Iske, E. Quak, M. Floater, Springer, 2002
10. J.H Ahn and Y.S. Ho, "Geometry Compression of 3-D Models Using Adaptive Quantization for Prediction Errors," *Proc. Picture Coding Symposium*, pp. 193-197, April 1999.