Thesis for Master’s Degree

A Study on View-Dependent Representation of 3-D Mesh Models Using Hierarchical Partitioning

Sung-Yeol Kim

Department of Information and Communications

Kwangju Institute of Science and Technology

2003
Abstract

As the demands of high quality visual service have increased recently from consumers, many researchers and companies have focused on the development of multimedia applications based on 3-D models, avatars and animations. However, it will be a difficult work to develop some applications related to 3-D mesh model transmission since they should accept the tremendous amount of transmission data for 3-D mesh models basically. Therefore, the study of efficient 3-D mesh model transmission scheme will be needed to manipulate those models in limited bandwidth channels.

In this thesis, we propose a new scheme for the transmission of 3-D mesh model information according to human viewing. The proposed scheme is able to say a version combining sequential and progressive mesh transmissions with view-dependent transmission. A view-dependent nonuniform mesh information can be transmitted through a network by setting higher priority to visible parts than invisible parts. The proposed system can be divided into three parts: hierarchical mesh partitioning to represent a mesh model, resolution decision of partitioned meshes and view-dependent 3-D mesh model transmission.

The first part is to represent a 3-D mesh model with a hierarchical partitioning structure. Hierarchical partitioning structure is obtained by doing mesh partitioning with previous partitioned meshes iteratively according to the specified number of initial partitioned mesh and level. In mesh partitioning, we use K-means algorithm for selecting start vertices. We also constitute multi-layers for progressive transmission based on the hierarchical mesh partitioning structure. Base layer and enhancement
layers exist in multi-layers for minimum transmission and additional transmission respectively. After representing a 3D mesh model to a hierarchical mesh partitioning structure, we determine the resolution of each partitioned mesh at the last level in hierarchical partitioning structure. We use the relationship between all normal vectors in each partitioned mesh and viewing position of viewer to decide the resolution. Finally, we transmit 3D mesh models by view-dependent transmission. Before transmitting 3D mesh models, we carry out partition mesh merging and partition mesh splitting operations in two viewing situations. In static viewing, that is, when the viewing position is not changed, partitioned mesh merging operation will be carried out to merge some partitioned meshes to upper level partitioned meshes in hierarchical mesh partitioning structure according to visibility. In dynamic viewing, that is, when the viewing position is changed by viewer to show invisible parts, partitioned mesh splitting operation will be carried out to determine the amount of additional transmitted mesh information to get refined 3D mesh models.

In this proposed system, there are two main contributions. One is that we proposed a relevant mesh model representation for view-dependent mesh model transfer which is called hierarchical mesh partitioning structure. The other is that the transmitted mesh information could be reduced through partitioned mesh merging and splitting operations which are view-dependent mesh partitioning conceptually.
Contents

Abstract (English) i
Abstract (Korean) iii
Acknowledgements v
List of Contents vii
List of Tables ix
List of Figures x

1 Introduction 1
  1.1 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
  1.2 Mesh Representation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
    1.2.1 Mesh Structure . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
    1.2.2 Mesh Topology . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
    1.2.3 3-D Representation Tools . . . . . . . . . . . . . . . . . . . . . . 5
  1.3 Organization of the Thesis . . . . . . . . . . . . . . . . . . . . . . . . 8

2 Review of Previous Works 10
  2.1 Mesh Simplification . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
    2.1.1 Vertex Clustering . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
    2.1.2 Incremental Mesh Simplification . . . . . . . . . . . . . . . . . . . 12
    2.1.3 Resampling . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
    2.1.4 Level-of-Detail . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
  2.2 Progressive Mesh . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
    2.2.1 Overview of Algorithm . . . . . . . . . . . . . . . . . . . . . . . . 17
    2.2.2 Application . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
  2.3 Mesh Partitioning . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
  2.4 View-dependent Progressive Mesh Coding . . . . . . . . . . . . . . . . 24

3 Proposed Scheme 29
  3.1 Hierarchical Mesh Partitioning . . . . . . . . . . . . . . . . . . . . . . 31
    3.1.1 Partitioning Procedure . . . . . . . . . . . . . . . . . . . . . . . . 32
3.1.2 Selection of Start Vertex ........................................ 33
3.1.3 Multi-Seed Traversal Technique ............................... 35
3.2 Multi-Layer Structure ............................................. 36
3.3 Resolution Decision ................................................ 37
3.4 Partitioned Mesh Merge and Split ............................ 41
   3.4.1 Partitioned Mesh Merge ..................................... 41
   3.4.2 Partitioned Mesh Split ..................................... 42
3.5 View-Dependent Mesh Transmission ........................... 44
   3.5.1 Static Viewing ............................................... 44
   3.5.2 Dynamic Viewing ........................................... 45

4 Experimental Results ................................................. 48
   4.1 Hierarchical Mesh Partitioning ............................... 48
   4.2 Static Viewing ................................................. 52
   4.3 Dynamic Viewing ............................................... 56
   4.4 Comparison of Results ........................................ 61

5 Conclusions and Future Work ....................................... 63

References ................................................................. 65
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>The resolution of each partitioned meshes of COW model in static viewing</td>
<td>53</td>
</tr>
<tr>
<td>4.2</td>
<td>The resolution of each partitioned meshes of MUSHROOM model in static viewing</td>
<td>54</td>
</tr>
<tr>
<td>4.3</td>
<td>The resolution of each partitioned meshes of BALL model in static viewing</td>
<td>54</td>
</tr>
<tr>
<td>4.4</td>
<td>The amount of transmitted mesh information in static viewing</td>
<td>56</td>
</tr>
<tr>
<td>4.5</td>
<td>The resolution of each partitioned meshes of MUSHROOM model in dynamic viewing</td>
<td>57</td>
</tr>
<tr>
<td>4.6</td>
<td>The resolution of each partitioned meshes of COW model in dynamic viewing</td>
<td>58</td>
</tr>
<tr>
<td>4.7</td>
<td>The resolution of each partitioned meshes of BALL model in dynamic viewing</td>
<td>59</td>
</tr>
<tr>
<td>4.8</td>
<td>The amount of additional transmitted mesh information in dynamic viewing</td>
<td>61</td>
</tr>
<tr>
<td>4.9</td>
<td>The resolution of each partitioned meshes of COW model in static viewing</td>
<td>62</td>
</tr>
</tbody>
</table>
List of Figures

1.1 Mesh Model Representation. ........................................... 4
1.2 Manifold and Non-manifold. ........................................... 5
2.1 Vertex Clustering Mesh Simplification. .............................. 12
2.2 Incremental simplification operations. .............................. 14
2.3 Mesh Resampling. ..................................................... 15
2.4 Level-Of-Detail. (a) Close viewing (b) Normal viewing (c) Far viewing. 16
2.5 Illustration of the edge collapse transformation .................... 17
2.6 (a)Sequence of edge collapses; (b)Resulting vertex correspondence ... 18
2.7 Multi-seed traversal scheme. ......................................... 22
2.8 Mesh Partitioning. .................................................... 23
2.9 Mesh Partitioning and Simplification in Yang’s algorithm. ......... 25
2.10 Vertex-based method. ............................................... 26
2.11 Partition-based method. ............................................. 27
2.12 Data organization for a mesh model and view-dependent transmission. 27
3.1 The frame work of the proposed system. ............................ 30
3.2 Algorithm for Hierarchical Mesh Paritioning ....................... 32
3.3 Selection of Start vertices ........................................... 35
3.4 The result of hierarchical mesh partitioning (K:2, L:3) ............ 36
3.5 Multi-layer structure on hierarchial partitioning structure. ........ 38
3.6 The resolution decision of partitioned mesh ....................... 40
Chapter 1

Introduction

As the demands for high quality visual service have increased from consumers recently, the interest of 3-D meshes which are widely used for 3-D object modelling has grown rapidly. To follow with this trend, the multimedia applications using 3-D mesh models have been appeared in so many fields such as games, movies and education tools. As network technologies have improved simultaneously, the 3-D mesh model transmission became an important issue in multimedia applications using 3-D mesh models. However, it is not easy for 3-D mesh models to transmit through a network since they require tremendous amount of data basically. Therefore, the study of efficient 3-D mesh model transmission scheme will be essential.

In this thesis, we propose a new scheme for progressive transmission of 3-D mesh information according to human viewing. A view-dependent nonuniform mesh information based on hierarchical mesh partitioning structure can be transmitted through a network by setting higher priority to visible parts. Using this technique, we are able to support the optimized quality of 3-D mesh models under limited transmission bandwidth. In addition to that, we can reduce the initial waiting time on web-based applications using 3-D mesh models since the visible parts and invisible parts are transmitted separately.
1.1 Motivation

We may have experienced some applications using 3-D mesh models such as 3-D avatar web-based applications. In these web-based applications, we should wait for a long time to get some information from these applications sometimes since 3-D mesh models occupy large parts in transmitted data. The waiting time can be an important factor to determine the application performance since consumers have no tolerance to wait. To reduce the waiting time, we are able to use view-dependent mesh transmission by transmitting the visible parts of 3-D mesh models to the receiver side first. In other words, the visible parts are transmitted with higher priority than invisible parts in 3-D mesh models. In addition to that, if we can transmit 3-D mesh models according to the degree of visibility, we may also reduce the amount of transmitted data. The progressive mesh representation will be suitable for this adaptive transmission.

We can adapt a mesh partitioning technique for 3-D mesh models to transmit which is called sequential mesh transmission. Mesh partitioning is a procedure which divides a given 3-D mesh model into several independent parts. If we transmit a mesh model with several partitioned meshes, it is useful in sense of storage and transmission of a mesh model since a spatial transmission error will not affect the entire model at the receiver side. Especially, view-dependent mesh partitioning will be a good answer for view-dependent transmission.
1.2 Mesh Representation

1.2.1 Mesh Structure

In general, a mesh model $M$ is simply a set of planar polygons in the three-dimensional Euclidean space $\mathbb{R}^3$. To represent a mesh surface, we can assume that the model consists of triangular faces entirely. Any non-triangular polygons can be triangulated by a pre-processing. Fundamentally, there are three types of information to describe the mesh surfaces. These are geometry information, connectivity information and photometry information. The geometry information describes vertex locations which are represented by three-dimensional floating vectors \( \{v_x, v_y, v_z\} \). The connectivity information describes the incidence relations among mesh elements which are vertices, edges and faces. A vertex is represented as an index in vertex incidence table. An edge \( \{e_{vid1}, e_{vid2}\} \) is represented as a set of two vertex indices. A face \( \{f_{vid1}, f_{vid2}, f_{vid3}\} \) is also represented as a set of three vertex indices. The photometry information includes surface normal vectors, colors, texture coordinates and other mesh properties attached to mesh elements. Fig. 1.1 shows the information of mesh representation.

We will use the following definition: a mesh model $M = (V, F)$ contains a list of vertices $V$ and a list of triangular face $F$. Here, we will leave out the photometry information definition since it will not be dealt in the rest of this thesis. The vertex list $V = (v_1, v_2, \cdots, v_r)$ is an ordered sequence which has the total number of vertex $r$. The face list $F = (f_1, f_2, \cdots, f_n)$ is also ordered sequence which has the total number of triangular face $n$. 
1.2.2 Mesh Topology

We divide mesh topology into two types commonly, manifold and non−manifold. We call a mesh manifold when it is topologically equivalent to a plane and a sphere. In other words, manifold meshes can not separated into two more planes or spheres. On the other hand, non−manifold meshes can be divided into two more planes or spheres, that is, two more manifold meshes[1][2].

There are two possible orientations for a face. These are clockwise and counterclockwise. We call the orientation of two adjacent faces compatible when the order of two vertices in common edge is different. A mesh is called orientable when all pairs of faces are compatible. In this thesis, we just deal with manifold and orientable meshes.

Fig. 1.2 shows a manifold mesh and a non-manifold mesh on the top part and also
describes a compatible mesh and an incompatible mesh on the bottom part.

1.2.3 3-D Representation Tools

VRML

VRML (Virtual Reality Modeling Language) is the standard language for describing 3-D objects and worlds composed of geometry and multimedia in a network environments. VRML focuses on Web applications such as CAD, engineering and scientific visualization, multimedia products, entertainment and educational offerings, architecture, medical imaging, molecular modeling, games, advertising of all varieties, and shared virtual worlds. The most exiting features of VRML is that it enables us to create dynamic worlds and sensory-rich virtual environments on the Internet, including the ability to [3]:
• Animate objects in our worlds, making them move:

VRML has a interpolator node in creating linear key-framed animation supporting interpolation in colors, coordinates and normal.

• Distributed Contents:

VRML supports World Wide Web and keeps some nodes that use URL’s connect the scene graph to the network. These nodes are:

1. Inline node: Add additional VRML content to create world composed of multiple VRML worlds.
2. Anchor node: Hyperlinks to other URL’s.
3. AudioCipe node: Generates sounds continuously.
4. ImageTexture node: Specifies texture in JPEG mapping attributes.
5. VideoTexture node: Specifies video contents in MPEG

• Allow users to interact with our worlds:

VRML uses several sensor nodes to implement specific commands from users (Environment sensors, pointing-device select sensors, and pointing-device drag sensors)

• Control and enhance worlds with scripts, small program us create to act on our VRML worlds.

We can use VRML to create interactive simulations that incorporate animation, motion physics, and real-time, multi-user participation. The virtual landscapes you
create can be distributed using the World Wide Web, displayed on another user’s computer screen, and explored interactively by remote users. As the Internet has being developed, VRML will be studied with many features and applications continuously[4].

OpenGL

OpenGL[5] is a software interface to graphics hardware. This interface consists of many commands that you use to specify the objects and operations needed to produce interactive three-dimensional applications. With OpenGL, you must build up your desired model from a small set of geometric primitives—points, lines, and polygons. It’s available on a variety of hardware platforms and operating systems, including Microsoft’s Window 95 and Window NT, IBM’s OS/2, DEC’s AXP and Open VMS. A sophisticated library that provides these features could certainly be built on OpenGL. The OpenGL Utility Library (GLU) provides many of the modeling features, such as quadric surfaces and NURBS curves and surfaces. GLU contains several routines that use lower-level OpenGL commands to perform such tasks as setting up matrices for specific viewing orientations and projections, performing polygon tessellation, and rendering surfaces. This library is provided as part of every OpenGL implementation. Several functions of OpenGL follows:

1. Texture mapping

The ability to apply an image to a graphics surface. This technique is used to rapidly generated realistic images without having to specify and excessive amount of detail.
2. Lightening effects

The ability to calculate the effects on the lightness of a surface’s color when different lighting models are applied to the surface from one or more light sources.

3. Smooth shading

The ability to calculate the shading effects that occur when light hits a surface at an angle and results in subtle color difference across the surface.

4. Material properties

The ability to specify the material properties of a surface

Because of these advantages, OpenGL is used with computer hardware which is designed and optimized for the display and manipulation of 3-D graphics.

1.3 Organization of the Thesis

In this thesis, we propose a new scheme for view-dependent mesh model transfer using hierarchical mesh partitioning. To represent a mesh model before transmission, we use a hierarchical mesh partitioning structure. After that, we will decide the resolution of each partitioned meshes by considering the relationship between normal vectors of vertices in partitioned meshes and the viewing position of viewer in receiver side. Finally, we will use the partitioned mesh merge operation and partitioned mesh split operation according to viewing situations for progressive mesh model transmission.

The thesis is organized as it follows.
In Chapter 2, we will briefly describe related works. It contains the overview of mesh simplification techniques and progressive mesh for multi-resolution representation. In addition to that, we will explore the concept of mesh partitioning which is called also mesh segmentation. Finally, a view-dependent transmission technique for a mesh model will be followed.

In Chapter 3, we will describe our proposed view-dependent mesh model transfer scheme using hierarchical mesh partitioning. It consists of five parts such as hierarchical mesh partitioning, multi-layer structure, resolution decision, partitioned mesh merge and split and view-dependent mesh transmission.

In Chapter 4, simulation results of the proposed system are presented. We will present results of tested mesh models in hierarchical mesh partitioning structure. Also, we will describe the results of our view-dependent mesh transmission in both static viewing situation and dynamic viewing situation.

In Chapter 5, we will summarize the main issues and recommends future works.
Chapter 2

Review of Previous Works

In this chapter, we will provide some previous works to help your understanding about the rest of this thesis. This chapter is composed of four parts structurally. First, we will examine for mesh simplification to obtain the concept of level-of-detail (LOD). To get the multi-resolution representation of 3-D models, we will talk about progressive mesh. After that, we will discuss mesh partitioning for error resilient transmission. Finally, we will explore a view-dependent transmission method for 3-D mesh model.

In 1995, Deering [6] suggested the concept of geometry data compression and connectivity data compression to reduce the 3-D mesh model data. After one year, Taubin and Rossignac[7] proposed another scheme to compress mesh models by using a triangle strips represented by a vertex spanning tree. To transmit 3-D mesh models over IP networks, it is advantageous for them to transmit in progressive manner. In 1996, Hoppe[8] first proposed the notion of progressive mesh which represents a mesh model into a continuous resolution based on the edge contraction and vertex split operations. Khodakovsky[10] proposed an alternative progressive coding scheme which uses the wavelet transform to represent a multi-layered mesh model. In addition to that, a view-dependent mesh simplification algorithm is proposed by Xia and Varshney[11]. In 1997, vertex-based view-dependent refinement[12] is presented by Hoppe.
2.1 Mesh Simplification

A mesh model consists of a fixed set of vertices and a fixed set of faces as we discussed on previous section. In general, the amount of mesh model data can be still larger than one of what we need to process in a practical manner. Therefore, we feel to reduce the mesh elements relevantly. A mesh simplification is a procedure to shrink the amount of mesh model data. In other words, mesh simplification describes a class of algorithms that transform a given mesh into another mesh with fewer faces, edges and vertices[16]. Suppose we have a mesh model $M$ and we would like to get an approximation mesh model $\hat{M}$. Basically, $\hat{M}$ should be as similar as possible to $M$ within defined quality criteria to preserve specific properties of the original mesh.

There are three conceptual approaches for mesh simplification. In this section, we will review these approaches and discuss the concept of level-of-detail(LOD).

2.1.1 Vertex Clustering

Vertex clustering is one of mesh simplification categories. The basic idea of vertex clustering is similar with vector quantization. For a given approximation tolerance, we first partition a mesh model into cells with diameter smaller than the tolerance. After that, we compute a representative vertex position for each cell. The representative vertex position can be calculated by a clustering algorithm. We can easily achieve mesh simplification by assigning a representative vertex position to all the vertices of its corresponding cell.

In vertex clustering algorithm, we can reduce the complexity very effectively since
it is able to change the topology of the given mesh model rapidly. In addition to that, the scheme guarantees a global approximation tolerance. However, it turns out that the approximation can fail to preserve the properties of original mesh sometimes[17]. To cover the draw-back, Algorri[18] and Cohen[19] proposed advanced vertex clustering algorithm by vertex grading. A weight is computed for each vertex according to its visual importance in advance. During mesh simplification using vertex clustering, the weight will make an effect to make the cells of original mesh model.

The vertex clustering algorithm is described in Fig.2.1

2.1.2 Incremental Mesh Simplification

Incremental mesh simplification algorithms remove one mesh vertex at a time unlike vertex clustering algorithms. We can obtain approximated meshes $\hat{M}$ from original meshes $M$ by the iterative procedure of incremental mesh simplification. The selection of the best candidate for removal is an important issue since approximated meshes $\hat{M}$
should preserve the topology of original meshes $M$. There are two user specified criteria for determining the best candidate, binary criterion and continuous criterion. Binary criterion uses the global approximation tolerance or to other minimum requirements like the minimum aspect ratio of triangles. Continuous criterion uses the local approximation tolerance by measuring the fairness of the mesh like small normal jumps between neighboring triangles. Whenever a vertex removal operation has been executed, we should reevaluate the quality criteria. To maintain the order of vertex removal, a *heap data structure* are usually used[20].

There are two topological operations in incremental mesh simplification. These are one vertex removal operation and edge collapse operation. One vertex removal deletes one vertex and its neighboring triangles. After removing the vertex, we retriangulate with the remained vertex information. edge collapse operation takes two adjacent vertices and contract the edge between them. After these topological operations, the number of vertices decreases by one and the number of triangles decreases by two consequently.

Fig.2.2 shows one vertex removal operation and edge collapse operation.

2.1.3 Resampling

Resampling techniques create new surface samples from the original mesh model $M$ and they are distributed on the faces of the original mesh model $M$ again. These samples are retriangulated for approximated meshes $\hat{M}$. Distributing the new samples can adapt the local vertex density to user specified requirement. However, alias error
caused by sampling can occur on the sharp part of original mesh model $M$.

Resampling techniques execute all faces of the given mesh. They place a new sample with a probability that is proportional to the area of the face or to some curvature dependent measure [21]. Because the number of faces is approximately twice the number of vertices in the original mesh, the resampling is very likely to place either one or none new sample in each original face. Resampling techniques are also used for remeshing which makes the valence of each vertex in original mesh regular.

**2.1.4 Level-of-Detail**

Fundamentally, mesh models are composed of a fixed number of vertices and faces. As a result, they only provide one fixed resolution representation of an object. However, it may not suitable for all the situations. The basic idea of level-of-detail (LOD) is to use simpler versions of an object as it makes less and less of a contribution to the
rendered images[22]. Fig.2.4 describes the concept of LOD. There is a cow mesh model that consists of 2903 vertices and 5804 faces. This representation can be used when the viewer is close to the mesh model. When cow is farther away, we do not need all 2903 vertices and 5804 faces. Instead, we can use a decimated model that has only 260 vertices and 500 faces. In this way, we can get a benefit in rendering time.

In general, LOD algorithms can be divided into three major parts, generation, selection and switching. LOD generation is the part that the mesh model is generated with different resolutions by mesh simplification methods. LOD selection chooses a level of mesh model based on some criteria such as the size of window. Finally, we need a process to change from one level to another. The process is named LOD switching.

2.2 Progressive Mesh

Hoppe proposed progressive mesh[8] in 1996. Progressive mesh is a new scheme for mesh model representation in order to transmit and to store arbitrary triangle meshes. Generally, progressive mesh is a continuous resolution representation of a model and
Figure 2.4: Level-Of-Detail. (a) Close viewing (b) Normal viewing (c) Far viewing.
used in smoothing level-of-detail approximations, progressive transmission, mesh compression and selective refinement[9]. To preserve the shape of models, progressive mesh adopt an optimization[23] procedure which measures the global distortion. The goal of this optimization is not just to preserve the geometry of the original mesh, but also importantly its overall appearance. In this section, we will deal with the overview of progressive mesh and its application.

2.2.1 Overview of Algorithm

Normally, transformations consists of edge collapse, edge split and edge swap. A single one of those transformations, edge collapse, is sufficient for effectively simplifying meshes. As shown in Fig. 2.5, an edge collapse transformation $\text{ecol}(\{v_s, v_t\})$ unifies 2 adjacent vertices $v_s$ and $v_t$ into a single vertex $v_s$. The vertex $v_t$ and two adjacent faces $\{v_s, v_t, v_l\}$ and $\{v_t, v_s, v_r\}$ vanish in the process. A positive $v_s$ is specified for the new unified vertex.

Figure 2.5: Illustration of the edge collapse transformation

Thus, an initial mesh $\hat{M} = M^n$ can be simplified into a coarser mesh $M^0$ by applying
a sequence of $n$ successive edge collapse transformations:

$$
(\hat{M} = M^n) \xrightarrow{\text{ecol}_{n-1}} \ldots \xrightarrow{\text{ecol}_1} M^1 \xrightarrow{\text{ecol}_0} M^0
$$

The particular sequence of edge collapse transformations must be chosen carefully, since it determines the quality of the approximating meshes $M^i$, $i < n$. The edges which represent outlines of models, must be preserved.

For example in Fig. 2.6, if one level of PM is $M^{i+1}$, the next level of $M^{i+1}$ can be represented as $M^i$ after edge collapse. In this process, two vertices $v_4$ and $v_7$ are unified into one vertex $v_4$. Thus total vertices of model in $M^{i+1}$ is less than that of model in $M^i$ by one vertex. Finally, 7 vertices are reduced to 3 vertices through $(i + 1)$ edge collapses.

Figure 2.6: (a) Sequence of edge collapses; (b) Resulting vertex correspondence

A vertex split transformation is described in Fig. 2.6 as inverse transformation of a vertex collapse. In vertex split transformation, vertex split variable $v_{\text{split}}(s, l, r, t, A)$ applied to original coarser mesh. As a result, one vertex $v_t$ is added and new faces $(v_l, v_s, v_t)$ and $(v_s, v_r, v_t)$ are created. The transformation also updates the attributes
of the mesh in the neighborhood of the transformation. This attribute information, denoted by $A$, includes the positions $v_s$ and $v_t$ of the two affected vertices, the discrete attributes $d_{\{v_s,v_t,v_l\}}$ and $d_{\{v_l,v_s,v_r\}}$ of the two new faces, and the scalar attributes of the affected corners $s_{(v,(v_s,v_t,v_l))}$, $s_{(v_t,(v_s,v_t,v_l))}$, $s_{(v_l,(v_s,v_t,v_l))}$ and $s_{(v_r,(v_t,v_s,v_r))}$.

An arbitrary triangle mesh can be represented as a simple mesh, $M^0$, and $n$ vertex split variables.

$$M^0 \xrightarrow{vsplit_0} M^1 \xrightarrow{vsplit_1} \ldots \xrightarrow{vsplit_{n-1}} (M^n = \hat{M})$$

(2.2)

where each record is parametrized as $vsplit_i(s_i, l_i, r_i, A_i)$.

We call $(M^0, \{vsplit_0, \ldots, vsplit_{n-1}\})$ a progressive mesh (PM) representation of $M$.

### 2.2.2 Application

- **Progressive Transmission**

  As we can see the meaning from the name, progressive meshes are a natural representation for progressive transmission. The basic mesh $M^0$ is transmitted first using a conventional uni-resolution format, followed by the stream of $vsplit_i$ variables. The receiver receives $M^0$ and update this base model with additional vsplit variables. The changes between base model and the updated model can be geomorphed and finally original mesh $\hat{M}$ is recovered exactly after receiving $n$ variables.

  This progressive transmission is similar to the display of images transmitted using progressive JPEG. With a slow communication line, a simple strategy is to display
the current mesh whenever the input buffer is found to be empty. With a fast communication line, we find that a good strategy is to display meshes whose complexities increase exponentially.

- **Mesh Compression**

   If we use progressive mesh in mesh compression, it is surprisingly space-efficient. For example, first, the locations of the vertex split transformations can be encoded concisely. Instead of storing all three vertex indices \((s_i, l_i, r_i)\) of \(vsplit_i\), one need only store \(s_i\) and approximately 5 bits to select the remaining two vertices among those adjacent to \(v_{s_i}\). Second, one can expect significant coherence in mesh attributes through each transformation. For instance, when vertex \(v_{s_i}^i\) is split into \(v_{s_i}^{i+1}\) and \(v_{m_0+i+1}^{i+1}\) from \(v_{s_i}^i\), and use delta-encoding to reduce storage.

   This progressive mesh scheme is new continuous-resolution scheme and it supports geomorphs, progressive transmission and compression In addition, this can be used in mesh simplification and multiresolution(continuous LOD) representation.

2.3 **Mesh Partitioning**

Mesh Partitioning is a procedure which divides a given 3-D mesh model into several independent parts. Mesh partitioning is also called mesh segmentation. If we transmit a mesh model with several partitioned meshes, it is useful in sense of storage and transmission of a mesh model since a spatial transmission error will not affect the entire model at the receiver side. Therefore mesh partitioning can make a mesh model to be robust on transmission error.
By using the mesh partitioning algorithm, the mesh model is divided into two types of elements, i.e. independent pieces and joint boundaries. Each independent piece is a part of the component, which can be encoded and decoded independently without affecting the processing of other pieces. There is a one-to-one mapping between vertices in the boundary of one piece and vertices in the boundary of another piece. We encode joint boundaries separately. Therefore, the independence of pieces can be realized with this information[24].

To do mesh partitioning, we first determine the number of partitioned meshes. The size of partitioned mesh is related with the target bit-error-rate(BER). Z. Yan proposed the adaptive piece size according to BER[15]. After deciding the number of partitioned meshes, the multi-seed traversal technique which provides an approach to achieve the independent segmentation is performed.

In initial process, we maintain lists as many as the number of partitioned meshes for traversing and we mark as unvisited to all vertices in a mesh model. After the end of process initialization, the unvisited vertices is traversed by a seed which is a vertex chosen from them. The traversed vertex index is marked as visited and it is pushed into the corresponding list for next seeds. We can traverse all vertices of a mesh model by iterative manner.

We can summarize the procedure of multi-seed traversal technique like following.

1. All the vertices in a given mesh model are marked as unvisited and make list sets \( \overrightarrow{L_i} \) for ordering of traversal.

2. To divide a connected component into \( N \) pieces, \( N \) vertices are first chosen as
starting seeds which are denoted by $\overrightarrow{S_i}$ and put the seeds into N polygon sets denoted by $\overrightarrow{P_i}$ for storing partitioned meshes. The seed vertices are marked as visited and are pushed into $\overrightarrow{L_i}$.

3. From $i = 1...N$ seed vertices, choose all unvisited vertices and traverse its neighboring vertices. If the vertices has not yet been traversed, put it into $\overrightarrow{L_i}$ and $\overrightarrow{P_i}$. Mark this vertices as visited.

4. For next traversal, Pop $i = 1...N$ seeds from $\overrightarrow{L_i}$ and put them into $\overrightarrow{S_i}$.

5. Repeat the above traversal until all vertices have been traversed.

After finishing the traversal procedure, we will make partitioned meshes with the information of $\overrightarrow{P_i}$. Fig. 2.8 shows a cow model divided into 4 partitioned meshes.
Figure 2.8: Mesh Partitioning.
2.4 View-dependent Progressive Mesh Coding

A view-dependent progressive mesh (VDPM) coding algorithm is proposed to facilitate interactive 3-D graphics streaming and browsing. The traditional mesh compression algorithm for 3-D mesh models can be divided into two parts which are the single resolution mesh coding and the progressive mesh coding. The single resolution mesh coding technique encodes a mesh model as a whole. On the other hand, the progressive mesh coding will transmit a mesh model progressively by updating mesh resolution from low to high continuously at the receiver. In the previous work, so many proposed work [12][10] are published. However, these algorithms do not use an assumption that only the front of 3-D mesh models are visible to the viewer in receiver side. In other words, the transmission of invisible parts is actually a waste of the limited bandwidth. Therefore, view-dependent progressive mesh (VDPM) is proposed. The basic idea of view-dependent progressive mesh is that the visible parts will be transmitted with higher priority that invisible parts in 3-D mesh models. Yang[14] proposed the view-dependent progressive mesh based on partitioning. We would like to explore Yang’s algorithm.

To enable view-dependent coding and transmission of a given mesh model, Yang’s algorithm divides a mesh model into several partitions and simplifies each partition independently of other partitions. The multi-seed traversal mesh partitioning technique are used for mesh partitioning. Yan and Kuo[15] proposed a mesh partitioning scheme, which is used for error-resilient mesh coding. However, the objective of the Yan and Kuo proposal is to enable the view-dependent compression and transmission,
Figure 2.9: Mesh Partitioning and Simplification in Yang’s algorithm.

not to facilitate the error concealment at the receiver side. Fig.2.9 describes the mesh partitioning and the mesh simplification procedures.

Let us assume that the original mesh is divided into three partitions as shown in the left figure, where partitioning boundaries are drawn with thick lines. The center figure in Fig.2.9 describes the base model by merging inner vertices of each partition into a single vertex. Finally, the right figure shows the connection information among partitioned meshes.

model adaptively to viewing positions, the visibility of each part should be determined. The accuracy and the complexity are two important factors in the visibility determination. There are in a trade-off relationship. Two methods are employed. The vertex-based method provides a complete and accurate list of visible vertices but requires a high computational complexity while the partition-based method requires fewer computations but lead to a larger error.
Vertex-based

Suppose that $V_O$ denotes the vertex to be judged and $\overrightarrow{V_E}$ denotes a vector pointing from $V_O$ to the viewer. In Fig.2.10, $\overrightarrow{N}$ is the normal vector of $V_O$. The angle $\theta$ between $\overrightarrow{V_E}$ and $\overrightarrow{N}$ is calculated by

$$\theta = \arccos\left\{ \frac{\overrightarrow{V_E} \cdot \overrightarrow{N}}{|\overrightarrow{V_E}| \cdot |\overrightarrow{N}|} \right\}$$

(2.3)

where $\cdot$ is the inner product operation. If $\theta$ is smaller than 90 degree then $V_O$ is declared as visible. Otherwise, it is declared as invisible.

Partition-based

For the partition-based method, the Gauss map[11] algorithm is adopted to determine the visibility. A Gauss map is a mapping of the unit normal vectors to the corresponding points on the surface of a unit sphere. All vectors within one subset can be represented by a cone. Fig.2.11 shows the decision method of appropriate visibility of each partition using Gauss map.

In view-dependent transmission, the client informs the server of its viewing posi-
Figure 2.11: Partition-based method.

Figure 2.12: Data organization for a mesh model and view-dependent transmission.
tions. Then, the server transmits each partition with an appropriate resolution. More specifically, a more visible partition is transmitted at a higher resolution, while an invisible partition is transmitted only its part in the base model.

In Fig.2.12 describes the data organization for a mesh model and an example of view-dependent transmission. The left figure shows that a mesh model consists of the base model and the layered data for each partitioned mesh. Base layer is the minimum transmitted layer without any situations such as network and client capability. However, the layered data is the additional transmitted data according to the calculated resolution of each partitions.
Chapter 3

Proposed Scheme

In previous chapters, we described some related works for helping you understand the proposed scheme which will be introduced. First we discussed the representation of 3-D mesh model. After that, we learned about mesh simplification methods which are related with Level-of-detail(LOD). Moreover, the progressive mesh representation of 3-D mesh models were followed. Finally, we talked about mesh partitioning techniques and the concept of view-dependent transmission. From now on, we will discuss our proposed scheme which is view-dependent progressive mesh model transfer using hierarchical partitioning structure. The proposed scheme is composed of four parts: the hierarchical mesh partitioning, resolution decision, partition merge or split and progressive transmission. Suppose that there is a VRML-based 3-D mesh model $M$. After the geometry information $V$ and connectivity information $F$ of 3D mesh models are extracted from the VRML-based format in preprocessing, we use these information for the input data of the proposed system. We do not extract the photometry information since it is out of range of our research. a 3-D mesh model will be divided into several independent pieces with a hierarchical structure which is called hierarchical mesh partitioning. Hierarchical mesh partitioning can be accomplished by repartitioning previous partitioned meshes. A 3-D mesh model can be represented by a hierarchical mesh partitioning structure. In mesh partitioning, we used multi-seed traversal tech-
nique. After representing a mesh model by a hierarchical mesh partitioning structure, we should determine the resolution of each partitioned mesh and then partitioned mesh merging operation and partitioned mesh splitting operation will be followed for view-dependent transmission. Finally, progressive transmission operates with corresponding resolutions. Fig. 3.1 shows the entire frame work of the proposed scheme.

There are two situations which are static viewing and dynamic viewing. In static viewing, the viewing position of viewer at the receiver side is not changed. In other
words, the viewer does not do any operations such as rotation, zooming and transition. On the other hand, the viewing position of viewer at the receiver side will be changed in dynamic viewing. The changed viewing parameter can be exchanged with messages between server and client.

3.1 Hierarchical Mesh Partitioning

Mesh Partitioning divides a given 3-D mesh model into several independent parts. When partitioned meshes are transmitted, it is useful in sense of storage and transmission of a mesh model since a spatial transmission error will not affect the entire model at the receiver side. Here, we proposed a new partitioning structure for representing a mesh model which is called hierarchical mesh partitioning. A mesh model has unique hierarchical mesh partitioning structure by setting the number of initial partitioned meshes and the number of level. The following definitions are used for developing this proposed scheme.

1. Boundary vertex : the vertex that lies on the partitioning boundaries

2. Inner vertex : the vertex that is not boundary vertex.

3. Start vertex : the vertex from which the mesh partitioning process starts.

4. Parent partitioned mesh : the partitioned mesh which will be divided into several partitioned meshes.

5. Child partitioned mesh : the partitioned mesh which is divided from a parent partitioned mesh.
3.1.1 Partitioning Procedure

Fig. 3.2 describes the entire algorithm for hierarchical mesh partitioning. The input data will be a mesh model which consists of geometry information and connectivity information. We should select the start vertices as many as the initial partitioned mesh. After that, we will do mesh partitioning process by using multi-seed traversal technique until the total number of partitioned meshes is larger than the number of partitioned meshes which is induced by using the relation between the number of the initial partitioned mesh $K$ and the specified number of level for hierarchical mesh partitioning.
partitioning \( L \). \( Curr_P \) and \( Curr_L \) mean the current number of partitioned meshes and the current level respectively. \# of \( Curr_L \) means the maximum number of partitioned meshes which is created at current level \( Curr_L \). For example, if the current level is 2 and the initial mesh partitioned meshes is 3 then \# of \( Curr_L \) is 9. \( L \) denotes the specified number of level. We have to initialize the current number of partitioned meshes to zero when the partitioning process is over at each level. After doing hierarchical mesh partitioning, the total number of partitioned mesh \( N \) is equal to \( K^L \).

### 3.1.2 Selection of Start Vertex

A VRML-based 3-D mesh model \( M = (V, F) \) is used for the input of hierarchical mesh partitioning,. \( V \) and \( C \) denote the geometry information and connectivity information of the mesh model respectively. we will use the connectivity information for mesh partitioning. As you know, the connectivity information is composed of the set of index of vertices. When we set a start vertex, we can find the neighboring vertices and the neighboring faces of the start vertex for mesh partitioning. In our algorithm, we set the number of initial partitioned mesh \( K \) which means the total number of child partitioned mesh from one parent partitioned mesh. To do mesh partitioning, we first select start vertices as many as the initial partitioned mesh \( K \) to start mesh partitioning. The selection of start vertices is important since they are related to the shape of partitioned mesh. We adopt the \( K \)-means algorithm (Ref) for selecting start vertices. This procedure consists of the following steps

1. Choose \( K \) initial center vertices \( Z_1(1), Z_2(1), ..., Z_K(1) \) by selecting arbitrary \( K \)
vertices in parent partitioned mesh.

2. At the k-th iterative step distribute the vertices \( \{X\} \) among the \( K \) cluster domains, using the relation.

\[
x \in S_j(k) \text{ if } \| x - z_j(k) \| < \| x - z_i(k) \| \quad (3.1)
\]

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

3. From the results of step 2, compute the new cluster centers \( z_j(k+1), \, j = 1, 2, ..., K \), such that \( K \) the sum of the 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 so that the performance index

\[
J_j = \sum_{x \in S_j(k)} \| x - x_j(k+1) \|^2, \, j = 1, 2, ..., K \quad (3.2)
\]

is minimized. The \( z_j(k+1) \) which minimizes this performance index is simply the sample mean of \( S_j(k) \). Therefore, the new vertices center is given

\[
z_j(k+1) = \frac{1}{N_j} \sum_{x \in S_j(k)} x, \, j = 1, 2, ..., K \quad (3.3)
\]

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 is terminated. Otherwise, go to step 2.

The behavior of \( K \)-means algorithm is influenced by the number of cluster centers specified. Therefore, the mesh partitioning from these start vertices can be operated in optimal manner.
Fig. 3.3 shows the result of the selection of start vertices by using maximum-distance algorithm (left) and $K$-means algorithm (right) when the number of the initial partitioned mesh $K$ is 3 at first level. We can notice that the optimal case is the use of the vertices chosen by $K$-means algorithm.

### 3.1.3 Multi-Seed Traversal Technique

After finishing the selection of start vertices, we will do mesh partitioning really. We adopt the multi-seed traversal technique for mesh partitioning. We introduced this scheme at chapter 2 briefly. In multi-seed traversal technique, we maintain lists as many as the number of partitioned meshes for mesh traversing and we mark as unvisited to all vertices in a mesh model in initial process. After that the unvisited vertices is traversed by a seed which is a vertex chosen from them. The first seed vertex of each partitioned mesh will be its start vertex. The traversed vertex index is marked as visited and it is pushed into the corresponding list for next seeds. we can do mesh
partitioning by iterative manner until the all vertices in a mesh model are marked as visited. Finally, we can separate the partitioned meshes from input mesh model by matching the connectivity information of input mesh model with the partition index of visited vertices. Fig.3.4 describes the result of hierarchical mesh partitioning of mushroom model with 2 initial partitioned mesh and 3 levels.

3.2 Multi-Layer Structure

After representing a mesh model into hierarchical mesh partitioning structure, we will do progressive mesh representation for each partitioned mesh. Each partitioned mesh at last level has one base layer and several enhancement layers. The base layer is for the minimum transmitted data to receiver and the enhancement layer is for
additional transmitted data according to the resolution of each partitioned mesh. As we mentioned in chapter 2, the progressive representation of a mesh model can be obtained by the inverse of mesh simplification. The base layer is simplified version of a mesh model and the enhancement layers are the removal information in mesh simplification process. We adopt the shape of base layer from Yang’s proposal[14].

When \(i\)-th partitioned mesh at last level is denoted by \(\hat{M}_i\), we can get base layer \(M^0_i\) by \(k\) times of edge contraction operations. Moreover, the information of \(k\) times of edge contraction is used for the enhancement layers.

Eq.3.4 shows the procedure for making the base layer of \(i\)-th partitioned mesh. We can see \(k\) times of edge contraction process.

\[
(\hat{M}_i = M^k_i) \xrightarrow{cont_i^{k-1}} ... \xrightarrow{cont_i^1} M^1_i \xrightarrow{col_i^0} M^0_i \tag{3.4}
\]

Eq.3.5 shows the reconstructed procedure from the base layer for \(i\)-th partitioned mesh. We can see \(k\)-times of edge split operations.

\[
M^0_i \xrightarrow{split_i^0} M^1_i \xrightarrow{split_i^1} ... \xrightarrow{split_i^{n-1}} (\hat{M}_i = M^k_i) \tag{3.5}
\]

3.3 Resolution Decision

After making multi-layer structure in hierarchical mesh partitioning, we should decide the resolution \(R\) of each partitioned mesh. According to the resolution \(R\), the amount of the transmitted data will be determined. To do this, we first determine the visible vertices and invisible vertices. It can be accomplished by checking the intersection angle \(\theta\) between the normal vector of each vertex and the viewing position.
Figure 3.5: Multi-layer structure on hierarchical partitioning structure.
\( \overrightarrow{V} \) of viewer. If the normal vector information is included in VRML-based model format then we can use this information. However, if there is no information about the normal vector then we should calculate the information with geometry information and connectivity information. \( n_j \) denotes the vertex normal vector which has the vertex index \( j \) and \( f_k \) denotes the face normal vector which has the face index \( k \). To calculate the \( n_j \), we first find the neighboring faces of \( j \) vertex. Let’s the neighboring faces as \( \{k_0, k_1, ..., k_{m-1}\} \) where \( m \) is the number of neighboring face of \( j \) vertex. \( f_k \) can be calculated by two vectors \( FV_1=\{x_1, y_1, z_1\} \) and \( FV_2 = \{x_2, y_2, z_2\} \).

\[
f_k = \frac{FV_1 \times FV_2}{\sqrt{|FV_1|^2 |FV_2|^2 - |FV_1 \cdot FV_2|^2}}
\]

(3.6)

\( n_j \) is calculate by following equation.

\[
n_j = \frac{1}{m} \sum_{i=0}^{m-1} f_{k_i}
\]

(3.7)

If the intersection angle \( \theta \) between \( n_j \) and \( \overrightarrow{V} \) is larger than 90°, the \( j \) vertex is invisible vertex. Otherwise, the \( j \) vertex is visible vertex. The \( \theta \) can be calculated by following equation.

\[
\theta = \arccos\left\{ \frac{\overrightarrow{V} \cdot n_j}{||\overrightarrow{V}|| |n_j|} \right\}
\]

(3.8)

There are two situations for deciding the resolution of partitioned mesh which are static viewing and dynamic viewing. In static viewing, we can notice that the resolution is the average of vertex resolutions which means the intersection angles \( \theta \) to reduce the initial transmitted data. On the other hand, the resolution in dynamic viewing is the minimum value among vertex resolutions to refine the transmitted mesh model quickly.

Fig.3.6 describes the resolution decision of a partitioned mesh.
We should determine the resolution of partitioned mesh $A$ in static viewing situation and dynamic viewing situation. First, we determine whether the all vertices in partitioned mesh $A$ is visible or not using Eq.3.8. After that, we should find the representative angle $\theta_A$ of partitioned mesh $A$. $\theta_A$ is the average of visible vertex angle in partitioned mesh $A$.

$$\theta_A = \min[\arccos\left(\frac{\vec{V} \cdot \vec{n}_j}{|\vec{V}||n_j|}\right)] \quad (3.9)$$

Therefore, the resolution $R_A$ of partitioned mesh $A$ in static viewing can be calculated by following equation.

$$R_A = R_{max}A \times \cos \theta_A \quad (3.10)$$

$R_{max}A$ denotes the maximum resolution of partitioned mesh $A$ which is the number of enhancement layer of $A$. Physically, Eq.3.9 means that the best quality for 3-D mesh models are supported.
3.4 Partitioned Mesh Merge and Split

Before view-dependent mesh transfer, we should adapt two operations to the hierarchical partitioned meshes according to viewing situation. These operations are partitioned mesh merge and partitioned mesh split. Basically, we use partitioned mesh merge operation in static viewing situation and partitioned mesh split operation in dynamic viewing situation.

3.4.1 Partitioned Mesh Merge

The partitioned mesh merge operation merges child partitioned meshes \( \{A_1, A_2, ..., A_K\} \) into their one-level upper parent partitioned mesh \( A \) according to the decision of their visibility where \( K \) is the number of the initial partitioned mesh. As we discussed at chapter 2, the partitioned mesh has joint boundaries and independent piece information. The joint boundaries are usually encoded twice. The smaller the size of partitioned meshes, the more joint boundaries will be transmitted redundantly. In Ref7, the minimum number of vertex in one partitioned mesh to be error-resilient transmission is 32. The boundaries information occupied about from 15% to 30%. By partitioned mesh merge, we can reduce the joint boundaries as we encode them just one time. After performing partitioned mesh merge, the resolution of merged mesh is different locally since the server will maintain the resolution for each child partitioned mesh before merging. Therefore, we can reduce the joint boundaries by merging while maintaining the resolution of each child partitioned mesh.

There are two methods in partitioned mesh merge. They are visible mesh merge and
invisible mesh merge. The visible mesh merge is performed among visible partitioned meshes. On the other hand, the invisible mesh merge is performed among invisible partitioned meshes. Fig.3.7 describes the partitioned mesh merge operation and the joint boundaries.

![Partitioned mesh merge](image)

Figure 3.7: The Partitioned mesh merge operation and joint boundaries

The partitioned mesh merge will be accomplished among the base layers in child partitioned meshes. In this procedure, the joint boundaries in each child partitioned meshes will be combined.

### 3.4.2 Partitioned Mesh Split

While the partitioned mesh merge is used for static viewing situation, the partitioned mesh split is used for dynamic viewing situation. This operation can be the inverse of partitioned mesh merge. However, the split operation is done conceptually not physically. In other words, the split meshes are not transmitted again since the initial mesh model is already transmitted to receiver in static viewing manner. The split operation is just for determining the amount of additional transmitted data to receiver.
As we mentioned in the subsection for resolution decision, we decide the resolution of partitioned meshes again to get the information of additional transmitted data. Suppose that the amount of additional transmitted data is $T_{add \ A}$ of partitioned mesh $A$. When $T_A$ denotes the amount of transmitted data to receiver for $A$ and $R_{curr \ A}$ denotes the current resolution of $A$, we can calculate $T_{add \ A}$ as following equation.

$$T_{add \ A} = R_{max \ A} \times R_{curr \ A} - T_A$$

(3.11)

If $T_{add \ A}$ is negative then we do not transmit additional data to receiver. Otherwise, we transmit as much as $T_{add \ A}$ to receiver additionally. There are two methods in partitioned mesh split. They are visible mesh split and invisible mesh split. The visible mesh split is performed when the splitting partitioned mesh i.e. the parent partitioned mesh is a visible partitioned mesh. On the other hand, the invisible mesh split is performed when the parent partitioned mesh is an invisible partitioned mesh. After splitting operation, the result should contain a visible partitioned mesh and an invisible partitioned mesh at least. Fig.3.8 show the partitioned mesh split operation.

Figure 3.8: The Partitioned mesh split operation
3.5 View-Dependent Mesh Transmission

The main idea of view-dependent mesh transmission is that the visible partitioned meshes are transmitted with higher priority than the invisible partitioned meshes. In other words, visible partitioned meshes are transmitted according to their resolution determined in advance, that is, base layer and enhancement layers are transmitted. On the other hand, the transmitted layers of invisible partitioned meshes are their base layers only. There are two situations which are static viewing and dynamic viewing: the initial viewing position of viewer is used for static viewing while the changed viewing position of viewer is used for dynamic viewing.

3.5.1 Static Viewing

In static viewing, we assume the viewing position of viewing at the side receiver will be not changed. In other words, there are no rotation, zooming and transition operations. We may have experienced some applications using static viewing situation such as 3-D avatar market software. We can also imagine some web-based applications. To start the web-based application, we should wait for a long time. Some user will be not tolerant for waiting them. Especially, Because 3-D based models need tremendous amount of storage basically, the waiting time will be still longer. In that situation, we can use view-dependent mesh transmission in static viewing manner by transmitting the visible initial mesh model first to receiver. After the visible initial mesh model, the text or image data will be followed to serve some pieces of information to user.

In static viewing situation, the partitioned mesh merge operation will be used to
transmit a mesh model. Fig.3.9 shows an example of view-dependent mesh transmission in static viewing. The used model is a mushroom mesh one which has hierarchical mesh partitioning structure with 2 initial partition mesh and 3 levels. We indicate sequential number for each level partitioned mesh. We can notice some partitioned mesh merge operation was performed. In Fig.3.9, The partitioned meshes at last level indicated by 1, 4, 5, 6 and 7 are visible partitioned meshes. The rest at last level, i.e. 0, 2 and 3 partitioned meshes are invisible partitioned meshes. The circled partitioned meshes will be transmitted to receiver. The thin circled partitioned meshes are visible partitioned meshes and the thick circled partitioned meshes are invisible partition meshes. As we can see in this figure, The visible partitioned merge operations are performed with partitioned meshes indicated by 4, 5, 6 and 7. In addition to that, the invisible partitioned merge operation is performed with partitioned meshes indicated by 2 and 3. Therefore, the entire transmitted data are 0 and 1 at 3 level, 1 at 2 level and 1 at 1 level. Here, 1 at level 3 and 1 at level 1 will be transmitted as much as their resolution and 0 at level 3 and 1 at level 2 will transmit only their base layer respectively.

3.5.2 Dynamic Viewing

In dynamic viewing, we assume the viewing position of viewing at the side receiver will be changed as time goes on. In other words, there are rotation, zooming and transition operations. The information of viewing position variation can be obtained by message exchange between sender and receiver. One case for dynamic viewing situation
Figure 3.9: View-dependent mesh transmission in static viewing can be the support about viewer requests in receiver side. After transmitting a mesh model in static viewing manner, we can notice the invisible parts will be distorted heavily since the visible parts have higher priority than invisible parts. Therefore, if the viewer in receiver side want to explore the invisible parts, we can not support high quality services. Another case for dynamic viewing situation can be additional transmission on 3-D web-based application after transmitting in viewing manner. It means the invisible parts will be transmitted later than the visible parts and other information such as texts and images.

In dynamic viewing situation, the partitioned mesh split operation will be used to transmit additional mesh data. Fig.3.10 shows an example of view-dependent mesh transmission in dynamic viewing.
Figure 3.10: View-dependent mesh transmission in dynamic viewing

The used model is a mushroom mesh one which has hierarchical mesh partitioning structure with 2 initial partition mesh and 3 levels. We indicate sequential number for each level partitioned mesh. We can notice some partitioned mesh split operation was performed. In Fig.3.10, we can notice that 1 at level 1 is split into 3 at level 2, 4 at level 3 and 5 at level 3 and also 1 at level 2 is split into 2 at level 3 and 3 at level 3. Therefore, After deciding whether additional data exists at each partitioned meshes or not, 4 at level 3 and 3 at level 2 will be retransmitted as much as the additional data.
Chapter 4

Experimental Results

In this thesis, we experimented with several mesh models. All tested mesh models have manifold and orientable topology as we mentioned in previous chapters. We will show the results of hierarchical mesh partitioning, the progressive transmission in static viewing situation and the progressive transmission in dynamic viewing situation respectively. We will compare the proposed scheme with Yang’s scheme in static viewing situation. We were just concerned about the amount of transmitted mesh information to the receiver in view-dependent progressive transmission manner.

4.1 Hierarchical Mesh Partitioning

As we mentioned in chapter 3, we represented several mesh models to hierarchical mesh partitioning structure. The tested models are COW model, MUSHROOM model and BALL model. COW model is composed by 2903 vertices and 5804 faces and we set that the initial partitioned mesh is 2 and the partition level is 5. MUSHROOM model consists of 226 vertices and 448 faces and we set the initial partitioned mesh is 2 and the partitioned level is 3. BALL model is composed by 762 vertices and 1520 faces and we set that the initial partition mesh is 3 and the partition level is 3. Fig.4.1, 4.2 and 4.3 show the hierarchical partitioning results of COW model, MUSHROOM model and BALL model respectively.
Figure 4.1: Hierarchical Partitioning of COW Model (2,5)
Figure 4.2: Hierarchical Partitioning of MUSHROOM Model (2,3)
Figure 4.3: Hierarchical Partitioning of BALL Model (3,3)
4.2 Static Viewing

We performed the transmission with COW model, MUSHROOM model and BALL model in static viewing situation. The results are based on the hierarchical partitioning structures of tested models in the previous section.

The resolutions of each partitioned mesh in COW model, MUSHROOM model and BALL model are described in Table 4.1, 4.2 and 4.3 when the viewing position of viewer is \((x = 0.0, y = 0.0, z = 1.0)\) in static viewing situation. PN and R denote the index and resolution of partitioned mesh. The notation \(Y\) and \(N\) mean visible partitioned mesh and invisible partitioned mesh respectively.

We adapted partitioned mesh merge operation to each tested model with corresponding resolution results. Fig.4.4, 4.5 and 4.6 show the static viewing results of COW model, MUSHROOM model and BALL model respectively. The left figure is the result of visible parts of viewer and the left figure is the result of visible parts of viewer. We are able to notice the viewing parts do not have any distortion but the invisible parts have some distortion because the amount of transmitted data in invisible parts is less than visible parts.

In COW model, 1 and 7 partitioned meshes at level 3, 0, 4, 9, 11 and 12 partitioned meshes at level 4 and 3, 12, 14 and 20 partitioned meshes level 5 were transmitted with their resolutions. However, 5, 8 and 13 partitioned meshes at level 4 and 2, 13, 15 and 21 partitioned meshes at level 5 were transmitted with their base layers by partitioned mesh merge operations.

In MUSHROOM model, 0 partitioned mesh at level 1 and 1 partitioned mesh at
Table 4.1: The resolution of each partitioned meshes of COW model in static viewing

<table>
<thead>
<tr>
<th>PN</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>0.98(Y)</td>
<td>0.98(Y)</td>
<td>-0.30(N)</td>
<td>0.87(Y)</td>
<td>0.58(Y)</td>
<td>0.87(Y)</td>
<td>0.58(Y)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PN</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>0.04(Y)</td>
<td>0.40(Y)</td>
<td>0.07(Y)</td>
<td>-0.10(N)</td>
<td>-0.02(N)</td>
<td>0.61(Y)</td>
<td>-0.04(N)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PN</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>0.10(Y)</td>
<td>-0.14(N)</td>
<td>-0.15(N)</td>
<td>-0.15(N)</td>
<td>0.03(Y)</td>
<td>0.04(Y)</td>
<td>0.69(Y)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PN</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>26</th>
<th>27</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>-0.20(N)</td>
<td>0.90(Y)</td>
<td>0.49(Y)</td>
<td>0.83(Y)</td>
<td>0.23(Y)</td>
<td>-0.14(N)</td>
<td>-0.19(N)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PN</th>
<th>28</th>
<th>29</th>
<th>30</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>-0.41(Y)</td>
<td>0.37(Y)</td>
<td>0.21(Y)</td>
<td>0.02(Y)</td>
</tr>
</tbody>
</table>

level 3 were transmitted with their resolutions and 1 partitioned mesh at level 2 and 0 partitioned mesh at level 3 were transmitted with their base layers.

In BALL model, 1 partitioned mesh at level 1, 6 partitioned mesh at level 2 and 21, 22, 24, 26 partitioned meshes at level 3 were transmitted with their resolutions. 1 partitioned mesh at level 1 and 23 and 25 partitioned meshes at level 3 were transmitted with their base layers.
Table 4.2: The resolution of each partitioned meshes of MUSHROOM model in static viewing

<table>
<thead>
<tr>
<th>PN</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>-0.12(N)</td>
<td>0.79(Y)</td>
<td>-0.22(N)</td>
<td>-0.11(N)</td>
<td>0.82(Y)</td>
<td>0.79(Y)</td>
<td>0.20(Y)</td>
<td>0.79(Y)</td>
</tr>
</tbody>
</table>

Table 4.3: The resolution of each partitioned meshes of BALL model in static viewing

<table>
<thead>
<tr>
<th>PN</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>1.00(Y)</td>
<td>0.98(Y)</td>
<td>0.98(Y)</td>
<td>0.95(Y)</td>
<td>0.70(Y)</td>
<td>0.89(Y)</td>
<td>0.98(Y)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PN</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>0.70(Y)</td>
<td>0.98(Y)</td>
<td>0.15(Y)</td>
<td>-0.58(N)</td>
<td>-0.15(N)</td>
<td>-0.30(N)</td>
<td>-0.70(N)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PN</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>-0.89(N)</td>
<td>-0.30(N)</td>
<td>-0.70(N)</td>
<td>-0.70(N)</td>
<td>0.80(Y)</td>
<td>0.58(Y)</td>
<td>0.70(Y)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PN</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>26</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>0.58(Y)</td>
<td>0.30(Y)</td>
<td>-1.15(N)</td>
<td>0.30(Y)</td>
<td>-0.30(N)</td>
<td>0.15(Y)</td>
</tr>
</tbody>
</table>
Figure 4.4: COW model: viewing Position \((x = 0.0, y = 0.0, z = 1.0)\), (a) visible parts, (b) invisible parts

Figure 4.5: MUSHROOM model: viewing Position \((x = 0.0, y = 0.0, z = 1.0)\), (a) visible parts, (b) invisible parts

Figure 4.6: BALL model: viewing Position \((x = 0.0, y = 0.0, z = 1.0)\), (a) visible parts, (b) invisible parts
Table 4.4: The amount of transmitted mesh information in static viewing

<table>
<thead>
<tr>
<th>Model</th>
<th>Original</th>
<th>Transmitted</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>V</td>
<td>F</td>
</tr>
<tr>
<td>COW</td>
<td>5804</td>
<td>2903</td>
<td>4362</td>
</tr>
<tr>
<td>MUSHROOM</td>
<td>488</td>
<td>226</td>
<td>372</td>
</tr>
<tr>
<td>BALL</td>
<td>1520</td>
<td>762</td>
<td>1010</td>
</tr>
</tbody>
</table>

Table 4.4 show the static viewing results of COW model, MUSHROOM model and BALL model respectively. We could reduce the transmitted data by view-dependent transmission about from 23% to 33%.

4.3 Dynamic Viewing

We performed the transmission with COW model, MUSHROOM model and BALL model in dynamic viewing situation. We changed the view position of view from $(x = 0.0, y = 0.0, z = 1.0)$ to $(x = 1.0, y = 0.0, z = -1.0)$.

The resolution of each partitioned meshes in COW model, MUSHROOM model and BALL model described in Table 4.6, 4.5 and 4.7 when the viewing position of viewer is changed. R denotes the resolution in dynamic viewing situation and D denotes the difference between the resolution in previous viewing and the resolution in changed viewing or no additional transmission. If the resolution in previous viewing is larger than the resolution in changed viewing, that is, D is No, we do not have to transmit additional data to corresponding partitioned mesh. Otherwise, we have to
Table 4.5: The resolution of each partitioned meshes of MUSHROOM model in dynamic viewing

<table>
<thead>
<tr>
<th>PN</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>0.74(Y)</td>
<td>0.13(Y)</td>
<td>0.90(Y)</td>
<td>0.79(Y)</td>
<td>0.11(Y)</td>
<td>0.59(Y)</td>
<td>0.87(Y)</td>
<td>0.96(Y)</td>
</tr>
<tr>
<td>D</td>
<td>0.74</td>
<td>No</td>
<td>0.90</td>
<td>0.79</td>
<td>No</td>
<td>No</td>
<td>0.67</td>
<td>0.17</td>
</tr>
</tbody>
</table>

some additional data to receiver as many as the difference resolution.

Fig.4.7, 4.8 and 4.9 show the dynamic viewing results of COW model, MUSHROOM model and BALL model respectively. The left figure is the result of previous viewing and the left figure is the result of dynamic viewing. We are able to notice that the new viewing parts do not have any distortion by additional transmission.

In COW model, the partitioned meshes for additional transmission are 6, 7, 8, 9, 10, 23, 28, 29, 30 and 31 partitioned meshes at level 5. In MUSHROOM model, the partitioned meshes for additional transmission are 0, 2, 3, 6 and 7 partitioned meshes at level 3. In BALL model, the partitioned meshes for additional transmission are 9, 10, 11, 12, 13, 14, 15, 16, 17, 22, 23, 24, 25 and 26 partitioned meshes at level 3. Table.4.8 shows the dynamic viewing results of COW model, MUSHROOM model and BALL model respectively.

- 57 -
Table 4.6: The resolution of each partitioned meshes of COW model in dynamic viewing

<table>
<thead>
<tr>
<th>PN</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>0.92(Y)</td>
<td>0.82(Y)</td>
<td>-0.21(N)</td>
<td>0.74(Y)</td>
<td>0.96(Y)</td>
<td>0.86(Y)</td>
<td>0.90(Y)</td>
</tr>
<tr>
<td>D</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>0.38</td>
<td>No</td>
<td>0.32</td>
</tr>
<tr>
<td>PN</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>R</td>
<td>0.09(Y)</td>
<td>0.74(Y)</td>
<td>0.74(Y)</td>
<td>0.45(Y)</td>
<td>-0.02(N)</td>
<td>0.18(Y)</td>
<td>-0.35(N)</td>
</tr>
<tr>
<td>D</td>
<td>0.05</td>
<td>0.34</td>
<td>0.67</td>
<td>0.45</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PN</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>R</td>
<td>-0.04(N)</td>
<td>-0.31(N)</td>
<td>-0.12(N)</td>
<td>-0.06(N)</td>
<td>-0.15(N)</td>
<td>-0.12(N)</td>
<td>0.21(Y)</td>
</tr>
<tr>
<td>D</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PN</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td>R</td>
<td>-0.08(N)</td>
<td>0.68(Y)</td>
<td>0.75(Y)</td>
<td>0.64(Y)</td>
<td>-0.30(N)</td>
<td>-0.30(N)</td>
<td>-0.23(N)</td>
</tr>
<tr>
<td>D</td>
<td>No</td>
<td>No</td>
<td>0.26</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PN</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>R</td>
<td>0.66(Y)</td>
<td>0.73(Y)</td>
<td>0.83(Y)</td>
<td>0.10(Y)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>0.25</td>
<td>0.36</td>
<td>0.62</td>
<td>0.08</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 4.7: The resolution of each partitioned meshes of BALL model in dynamic viewing

<table>
<thead>
<tr>
<th>PN</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>-0.44(N)</td>
<td>-0.62(N)</td>
<td>-0.40(N)</td>
<td>0.04(Y)</td>
<td>0.58(Y)</td>
<td>0.20(Y)</td>
<td>-0.02(N)</td>
</tr>
<tr>
<td>D</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PN</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>R</td>
<td>0.51(Y)</td>
<td>0.58(Y)</td>
<td>0.99(Y)</td>
<td>0.99(Y)</td>
<td>0.99(Y)</td>
<td>0.73(Y)</td>
<td>0.71(Y)</td>
</tr>
<tr>
<td>D</td>
<td>No</td>
<td>No</td>
<td>0.84</td>
<td>0.99</td>
<td>0.99</td>
<td>0.73</td>
<td>0.71</td>
</tr>
<tr>
<td>PN</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>R</td>
<td>0.95(Y)</td>
<td>0.90(Y)</td>
<td>0.71(Y)</td>
<td>0.91(Y)</td>
<td>-0.63(N)</td>
<td>-0.27(N)</td>
<td>-0.01(N)</td>
</tr>
<tr>
<td>D</td>
<td>0.95</td>
<td>0.90</td>
<td>0.71</td>
<td>0.91</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PN</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>0.01(Y)</td>
<td>0.61(Y)</td>
<td>0.32(Y)</td>
<td>0.73(Y)</td>
<td>0.34(Y)</td>
<td>0.23(Y)</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>No</td>
<td>0.31</td>
<td>0.32</td>
<td>0.43</td>
<td>0.34</td>
<td>0.08</td>
<td></td>
</tr>
</tbody>
</table>
Figure 4.7: COW model: viewing position \((x = 1.0, y = 0.0, z = -1.0)\), (a) before viewing position was changed, (b) after viewing position was changed.

Figure 4.8: MUSUROOM model: viewing position \((x = 1.0, y = 0.0, z = -1.0)\), (a) before viewing position was changed, (b) after viewing position was changed.

Figure 4.9: BALL model: viewing position \((x = 1.0, y = 0.0, z = -1.0)\), (a) before viewing position was changed, (b) after viewing position was changed.
Table 4.8: The amount of additional transmitted mesh information in dynamic viewing

<table>
<thead>
<tr>
<th>Model</th>
<th>Static Viewing</th>
<th>Dynamic Viewing</th>
<th>Addition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F</td>
<td>V</td>
<td>F</td>
</tr>
<tr>
<td>COW</td>
<td>4362</td>
<td>2189</td>
<td>4898</td>
</tr>
<tr>
<td>MUSHROOM</td>
<td>372</td>
<td>172</td>
<td>414</td>
</tr>
<tr>
<td>BALL</td>
<td>1010</td>
<td>507</td>
<td>1468</td>
</tr>
</tbody>
</table>

4.4 Comparison of Results

We compared the proposed system with Yang’s method. The compared 3-D mesh model was COW model. Table 4.9 shows the compared results of two method and Fig 4.10 also shows the result of each method. Conceptually, proposed system can be view-dependent mesh partitioning since the partitioned meshes are combined together according to visibility. In other words, the visible parts are merged together and the invisible parts are also merged together. However, we maintained the resolution of previous partitioned meshes after merging operations and transmitted corresponding resolution to receiver. In addition to that, we could reduce the common boundary information. Basically, the common boundary information is transmitted twice because it represent the border of each partitioned meshes. By doing partitioned mesh merge operations, we could reduce the redundancy of common boundary information. we could reduce the transmitted mesh information more from 5% to 10% than Yang’s method.
Table 4.9: The resolution of each partitioned meshes of COW model in static viewing

<table>
<thead>
<tr>
<th>Test Model</th>
<th>Original Model</th>
<th>Yang’s scheme</th>
<th>Proposed Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F  V</td>
<td>F  V</td>
<td>F  V</td>
</tr>
<tr>
<td>COW</td>
<td>5804 2904</td>
<td>4960 2429</td>
<td>4362 2189</td>
</tr>
</tbody>
</table>

Figure 4.10: Comparison with Yang’s scheme
Chapter 5

Conclusions and Future Work

In this thesis, we proposed a new scheme for view-dependent progressive transmission of 3-D mesh models. To transmit 3-D mesh models efficiently, we represented these models into hierarchical mesh partitioning structures. After that, we determined the resolution of each partitioned mesh at the last level in hierarchical partitioning structure. Finally, we transmitted these models progressively according to the viewing position of viewer at the receiver side.

Usually, it takes a long time to transmit 3-D mesh models since they have tremendous amount of data. In this thesis, we could reduce the amount of transmitted data by focusing on the visible parts at the receiver side. In addition to that, a mesh partitioning technique was used for transmission of 3-D mesh models and we transmitted 3-D mesh models efficiently by carrying out partitioned mesh merging operation and partitioned mesh splitting operation relevantly. We expect our scheme to be useful in many web-based multimedia applications using 3-D mesh models since they are transmitted according to the degree of visibility. Although we just limited a mesh model transmission in this thesis, the scheme can be extended to 3-D animation transmission.

We are able to consider some further works to improve performance of view-dependent transmission scheme. First, we assumed that the connectivity information in 3-D mesh models are not changed during preprocessing. However, if we change
the mesh information appropriately in preprocessing, we may have a chance to get more efficient multi-layer representations. One candidate for preprocessing is re-mesh approach which make the valence of all vertices in a 3-D mesh model equal. After re-meshing, we are able to apply the signal transformation such as DCT and wavelet transform to 3-D mesh models to making multi-layer structure. Second, there is base model definition issue. The skillful definition of base model helps 3-D mesh model transmission when the network situation is bad since base model can be the minimum service for users. Finally, we are able to consider error concealment. Error concealment techniques are used to recover the lost information. By extending 2-D error concealment techniques into 3-D mesh models, we may expect good performance when the lost packets including 3-D mesh model data occur during transmission.
References

1. J. Rossignac and Gardoze, “Matchmarker:Manifold Breps for non-manifold r-sets,”
   Tech. Rep. GIT-GVU-00-03 GVU Center, Georgia Inst. of Tech.

2. A. Gueziec and G. Taubin, “Converting sets of polygons to manifold surfaces by

3. G. Taubin, W. Horn, F. Lazarus and J. Rossignac,”Geometric Coding and

4. J. Hartman and J. Wernecke, ”The VRML 2.0 Handbook,” Addison-Wesley Pub-


7. G. Taubin and J.Rossignac,”Research Report RC-20340, IBM Research Division,
1996.


of Arbitrary Polygonal Meshes,” Department of Computer Sciences and TICAM
University of Texas


