Communication of 3D Graphic Models

COMMUNICATION OF 3D GRAPHIC MODELS

BY

INSU PARK, B.Sc., M.A.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

Master of Applied Science (2006) McMaster University

(Electrical & Computer Engineering) Hamilton, Ontario, Canada

TITLE:	Communication of 3D Graphic Models
AUTHOR:	Insu Park B.Sc.(Electrical Engineering) Chosun University, Kwang-Ju, Korea M.A.Sc.(Electrical Engineering) Sogang University, Seoul, Korea
SUPERVISOR:	Dr. Shahram Shirani Dr. David W. Capson
NUMBER OF PAGES:	xii, 95

Abstract

A new area-based mesh simplification algorithm is described. The proposed algorithm removes the center vertex of a polygon which consists of $n \geq 3$ faces and represents that polygon with $n - 2$ faces. A global search method is adapted that iteratively determines which vertex is to be removed using the proposed area-based distortion measurement. Although the global search method requires more computations compared to a local search method, it guarantees better quality of approximation. Various re-triangulations are also considered to improve the perceptual quality of the final approximation.

From multiple re-triangulations, one with minimum distortion is selected to represent the original mesh. Experimental results demonstrate the performance of the proposed algorithm for data reduction while maintaining the quality of the rendered objects. The performance of multiple description decoder when not all descriptions are available depends on the decoding strategy. By approximating lost description the distortion can be reduced. When decoder reconstructs input source without having all descriptions different methods exist to approximate the lost description. We proposed two side decoding algorithms. The proposed side decoders are based on diagonal element and the probability of input source. When low bit-rate and complicated index assignment matrix are used the side decoder

based on probability of input source is recommendable. To approximate the lost description we compare the performance of standard decoding method with the performance of proposed methods. A trade-off between the performance of decoder and computational complexity exists.

An error concealment algorithm is proposed based on flow of facial expression to improve communication of animated facial data over a limited bandwidth channel with error. Facial expression flow is tracked using dominant muscles which are those with maximum change between two successive frames. By comparing the dominant muscle data with the predetermined expression information table, facial expression flow is determined. The receiver uses linear interpolation and the information on facial expression flow to interpolate the erroneous facial animation data. For objective comparison, a distortion measurement tool, which compares two 3D objects based on point-to-point difference, is introduced. Experimental results are provided to show that the proposed error concealment method improves the quality of an animated face communications.

Acknowledgements

I would like to sincerely thank and acknowledge my supervisor, Dr. Shahram Shirani and Dr. David W. Capson, my supervisor, for their continual support, technical and personal guidance, and trust. I am grateful to them for providing me many opportunities.

In addition I would like to thank Dr. Shahin Sirouspour and Dr. Aleksandar Jeremic who served as the members of my supervisory committee for their valuable comments. I would also like to thank Dr. Steve Hranilovic for making a number of helpful suggestions. I acknowledge gratefully Dr. Cameron M. Crowe who served as my English teacher for many years.

Sincere thanks go to my friends and colleagues at Multimedia Signal Processing and Computational Vision Lab for their helps. I also thank Cheryl Gies, Adam Marianski, Terry Greenlay, Cosmin Coroiu, and Helen Jachna for their heartwarming helps. To anyone else who had patience with me while I completed my work I would like to express my gratitude.

Last but not least, I want to thank my family who always support and encourage me through my life. Their understanding and patience have made it possible to pass all the steps. Without their supports, I can not imagine present myself.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Representing 3 dimensional (3D) objects using computer graphic models and their applications have received considerable interest in various application areas. Data acquisition tools and range scanners for 3D easily generate millions of polygons for a simple object and require substantial storage. It may be redundant to represent a simple object in too much detail with excess data. Also communication of 3D graphic model is essential in many multimedia applications such as videoconferencing, distance learning and virtual reality. All of these applications demand rendering of a realistic and comfortable 3D graphic model.

In (3D) representation, 3D data consists of two data sets: (1) geometrical data describing information of each vertex and (2) topological data which is a set of vertex members of every polygon. The structure of 3D object (S) can be expressed as $S = (\mathcal{V}, \Psi)$ where $\mathcal{V} = \{v_1, v_2, v_3, \cdots v_n\}$ is n geometrical data ($v_n = \{x_n, y_n, z_n\}$) and

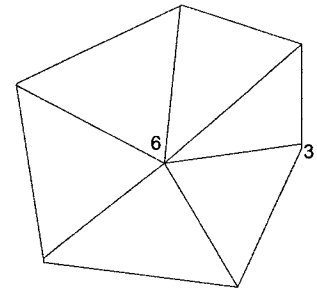


Figure 1.1: Degree of vertices

$\Psi = \{\psi_1, \psi_2, \psi_3, \cdots \psi_m\}$ is m topological data ($\psi_i = \{v_1, v_2, v_3, \cdots v_k\}$). If ith polygon (ψ_i) consist of k vertices we call ith polygon as $k-$gon. Each topological data consists of more than 3 geometrical data elements. Since triangle is the elementary unit of a polygon we assume that 3D objects are represented by a set of triangles.

The degree of a vertex is defined as the number of edges which touch the vertex. Since all polygons consist of triangles, the degree of the center vertex of a polygon represent how many faces the polygon has. Figure 1.1 illustrates the degree of vertices in a polygon. If the function $\rho(v)$ denote the degree of a vertex v then the total number of edges ($e(S)$) of a 3D object (S) is

$$e(S) = \frac{1}{2} \sum_{i=1}^{n} \rho(v_i) \tag{1.1}$$

where n is the total number of vertices forming the 3D object.

A boundary vertex is a vertex located on the boundary of the object. Verifying boundary vertex is enough with checking whether all faces formed by that vertex or not. If all these faces are connected to each other, the vertex is not a boundary vertex. Otherwise the vertex is a boundary vertex. The concept of boundary vertex

is important in mesh simplification. If a boundary vertex is removed without care, the shape of object can be easily lost.

Three major methods for mesh simplification are vertex removal, vertex clustering and edge collapse. Schroeder [1] introduced mesh simplification based on vertex removal. The vertex removal method chooses one vertex for removal, removes all of the adjacent faces which contain the vertex being removed and retriangulates the polygon. Thus, in every iteration one vertex and two faces are removed from original mesh. In vertex clustering algorithms, a bounding box is employed to merge some vertices [2]. All of the vertices included in the same bounding box are represented with one vertex. Edge collapse methods delete one edge and merge two vertices to one vertex in every iteration. Kobbelt [3] developed mesh decimation using half-edge collapse. In some mesh simplification methods, an edge swap function is adopted to improve the approximation [4].

The common issue of these methods is which candidate vertex or edge must be removed at each iteration. If we remove an element of a polygon from the original geometrical data or change the topological structure, there will be distortion. Thus, when we remove geometrical data or change topological structure, we must consider that distortion. The first candidate must be a vertex or an edge that introduce minimum distortion.

After representing 3D graphic model with simplified data set, communication 3D graphic model over error prone channel is another issue. 3D graphic model without motion can be transmitted via different physical channel for error concealment. Also the motion information of moving 3D graphic model can be investigated to reduce the effect of channel error.

1.2 Contribution

In this thesis, we investigated 3D object in two different points. 3D mesh simplification is a classic issue to reduce data size for 3D object. Whenever we reduce a data distortion exist. The objective of 3D mesh simplification is reducing the data requiring to represent a 3D object with small distortion.

After reduce the data of 3D object, communication over error prone channel is another issue. For communication of 3D object we deal with both moving 3D object and 3D object without motion. We propose error concealment method based on linear interpolation for 3D object with motion. Also multiple description, which is investigated for random signal and image, is applied for 3D object without motion.

The contents of Chapter 2, 3 and 4 are based on published journal and conference papers. The earlier works of mesh simplification is published in [5]. More works of mesh simplification based on surface are is in [6]. Our mesh simplification algorithm is explained in journal paper in detail [7]. The error concealment method using motion information of human face model is published in [8].

1.3 Structure of Thesis

We describe some different methods for quantifying this difference in Chapter 2. Chapter 3 deals with 3D mesh simplification. The area based error criterion is studied in this Chapter. The issue of communication of 3D object over error prone channel is studied in Chapter 4. In this Chapter we introduce multiple description coding and linear interpolation for error concealment tool. This thesis finalized with conclusion in Chapter 5

Chapter 2

Difference of two 3D models

2.1 Distortion measure tool

Since the visible shape of 3D object depends on the view point it is not easy to define the difference between two 3D objects. Simple approach for this problem is calculating some difference values based on different view point and sum those difference values. This approach consider 3D object as the sum of $2\frac{1}{2}$D models. Let us consider the appearance of a model S under viewing conditions ζ is determined by the image $I^{\zeta}(S)$. Then we can say the two models S_1 and S_2 appear identical in the view point ζ if their corresponding 2D images $I^{\zeta}(S_1)$ and $I^{\zeta}(S_2)$ are identical. If $I^{\zeta}(S_1)$ and $I^{\zeta}(S_2)$ are not same, the difference between two 3D objects under viewing condition ζ, $\mathfrak{D}^{\zeta}(S_1, S_2)$ can be considered with the difference between them using the sum of squared difference:

$$\begin{aligned} \mathfrak{D}^{\zeta}(S_1, S_2) &= \|I^{\zeta}(S_1) - I^{\zeta}(S_2)\|^2 \\ &= \textstyle\sum_x \sum_y \|I^{\zeta}(S_1; x, y) - I^{\zeta}(S_2; x, y)\|^2. \end{aligned} \tag{2.1}$$

Using this difference function the total difference of two 3D objects S_1 and S_2 is defined by summing $\mathfrak{D}^{\zeta}(S_1, S_2)$:

$$\mathfrak{E}(S_1, S_2) = \sum_{\zeta} \mathfrak{D}^{\zeta}(S_1, S_2). \tag{2.2}$$

There are other approaches calculating the difference of tow 3D objects directly. *Metro* is a distortion measurement tool that can quantify the difference between two 3D objects. *Metro* which is based on the Hausdorff distance was introduced by Cignoni [9]. Given a point set ($\Gamma(\mathcal{S})$) on the object $\mathcal{S}$ the distance from one point v to $\mathcal{S}$ is

$$d(\mathcal{S}, v) = \min_{w \in \Gamma(\mathcal{S})} \|v - w\|, \tag{2.3}$$

where $\|.\|$ denotes Euclidean distance. Then one-sided distance between two objects $\mathcal{S}_1, \mathcal{S}_2$ is

$$d(\mathcal{S}_1, \mathcal{S}_2) = \max_{v \in \Gamma(\mathcal{S}_1)} d(\mathcal{S}_2, v). \tag{2.4}$$

Then, the Hausdorff distance between two 3D objects $\mathcal{S}_1$ and $\mathcal{S}_2$ is

$$\mathfrak{E}_{max}(\mathcal{S}_1, \mathcal{S}_2) = \max\left(d(\mathcal{S}_1, \mathcal{S}_2), d(\mathcal{S}_2, \mathcal{S}_1)\right). \tag{2.5}$$

Also the average distance between two 3D objects $\mathcal{S}_1$ and $\mathcal{S}_2$ is

$$\mathfrak{E}_{avg}(\mathcal{S}_1, \mathcal{S}_2) = \frac{1}{\mathfrak{A}(\mathcal{S}_1) + \mathfrak{A}(\mathcal{S}_2)} \left(\Psi(\mathcal{S}_1, \mathcal{S}_2) + \Psi(\mathcal{S}_2, \mathcal{S}_1)\right) \tag{2.6}$$

where $\mathfrak{A}(\mathcal{S})$ denotes the area of the object $\mathcal{S}$ and sum of distance function $\Psi(.,.)$ is defined as

$$\Psi(\mathcal{S}_1, \mathcal{S}_2) = \int_{\Gamma(\mathcal{S}_1)} d(\mathcal{S}_2, v) dv.$$

Since the computational complexity of $\mathfrak{E}_{avg}$ is high usually surface $\mathcal{S}_1$ and $\mathcal{S}_2$ are sampled. Let consider two sets $\mathcal{X}_1 \subset \Gamma(\mathcal{S}_1)$ and $\mathcal{X}_2 \subset \Gamma(\mathcal{S}_2)$ containing n_1 and n_2 sample point, then $\mathfrak{E}_{avg}$ can be expressed as [10]

$$\mathfrak{E}_{avg}(\mathcal{S}_1, \mathcal{S}_2) = \frac{1}{n_1 + n_2} \left(\sum_{v \in \mathcal{X}_1} \mathfrak{D}(\mathcal{S}_2, v) + \sum_{v \in \mathcal{X}_2} \mathfrak{D}(\mathcal{S}_1, v) \right) \tag{2.7}$$

The average distance can be extended to define the distance between two surfaces $\mathcal{S}_1$ and $\mathcal{S}_2$ using *root mean square error*. The average distance based on root mean square error (rms) is

$$\begin{aligned} &\mathfrak{E}_{rms}(\mathcal{S}_1, \mathcal{S}_2) \\ &= \frac{1}{\mathfrak{A}(\mathcal{S}_1)+\mathfrak{A}(\mathcal{S}_2)} \left(\Psi^2(\mathcal{S}_1, \mathcal{S}_2) + \Psi^2(\mathcal{S}_2, \mathcal{S}_1) \right)^{\frac{1}{2}} \\ &= \frac{1}{n_1+n_2} \left(\sum_{v \in \mathcal{X}_1} d^2(\mathcal{S}_2, v) + \sum_{v \in \mathcal{X}_2} d^2(\mathcal{S}_1, v) \right)^{\frac{1}{2}}. \end{aligned} \tag{2.8}$$

These three distortion measurement ($\mathfrak{E}_{max}$, $\mathfrak{E}_{avg}$ and $\mathfrak{E}_{rms}$) are used later in this thesis to define $PSNR_{max}$, $PSNR_{avg}$ and $PSNR_{rms}$, respectively. Also, in terms of computation complexity, a more efficient algorithm was introduced by Aspert [11].

2.2 Peak signal to noise ratio

Thus, with the existence of channel errors and possible change in a 3D object due to simplification, the reconstructed 3D object ($\bar{\mathcal{S}}$) can be different from the original ($\mathcal{S}$). The distortion between ($\bar{\mathcal{S}}$) and ($\mathcal{S}$) can be quantified by peak signal to noise ratio (PSNR). The PSNR of reconstructed 3D object is defined as [12]:

$$\begin{aligned} \text{PSNR}_{\text{max}} &= -20 \log \frac{\mathfrak{E}_{\text{max}}(\mathcal{S}, \bar{\mathcal{S}})}{\mathfrak{L}(\mathcal{S})} \\ \text{PSNR}_{\text{avg}} &= -20 \log \frac{\mathfrak{E}_{\text{avg}}(\mathcal{S}, \bar{\mathcal{S}})}{\mathfrak{L}(\mathcal{S})} \\ \text{PSNR}_{\text{rms}} &= -20 \log \frac{\mathfrak{E}_{\text{rms}}(\mathcal{S}, \bar{\mathcal{S}})}{\mathfrak{L}(\mathcal{S})} \end{aligned}$$

where $\mathfrak{E}_{\text{max}}$, $\mathfrak{E}_{\text{avg}}$ and $\mathfrak{E}_{\text{rms}}$ are maximum, average and root mean square error defined by Equations (2.5), (2.7) and (2.8), respectively. $\mathfrak{L}(\mathcal{S})$ denote diagonal length of bounding box enclosing the object $\mathcal{S}$.

Chapter 3

Mesh simplification of 3D models

3.1 Introduction

In recent years, representing 3D objects and their applications have received considerable interest in various areas. One approach in representation of 3D surfaces is by polygonal meshes. Data acquisition tools and range scanners for 3D easily generate millions of polygons for a simple object and require substantial storage. It may be redundant to represent a simple object in too much detail with excess data. Although graphic acceleration techniques have been developed, the speed of rendering and transmission of 3D objects depends primarily on the size of data set.

Thus, for transmission and storage applications, it is necessary to reduce the size of the 3D data which is required to represent an object. This can be achieved with two different approaches. One is using traditional compression methods such as quantization, wavelet and statistical encoding [13]. Khodakovsky [14] used a wavelet transform and a zero tree coder to develop a progressive compression

algorithm for semi-regular meshes. The other option is to employ a mesh simplification method. A mesh simplification method, using original mesh structures (geometrical and topological data), generates a new mesh structure having less data than that of the original. Mesh simplifications were introduced to simplify mesh structures by eliminating elements of polygons (vertex, edge, face) or by changing the topological structure. When a very simple object is represented with a large amount of data, mesh simplification is more effective than traditional compression methods.

Mesh simplification introduces distortion between the original and the approximation. When a mesh simplification algorithm is designed, an important issue is the selection of a vertex or face to be deleted. Choosing optimal vertices, (the vertices which produce minimum distortion) guarantees minimization of the distortion between the original and final approximation. After a vertex is removed, the mesh must be re-triangulated. Since the distortion also depends on how re-triangulation is performed, re-triangulating the modified area is another important issue in mesh simplification.

Choosing the optimal vertex depends on the distortion measure. Different distortion measure can yield different vertices as the optimal vertex. In addressing this problem, we describe an area based distortion measure. In proposed method, a priority list of all vertices of a mesh which can be a candidate for removal is formed. The priority is based on the amount of distortion that the removal of a particular vertex will introduce. The priority list is employed to reduce computational complexity. Since the required computation complexity is an exponential function of the number of vertices of the original mesh, it is time consuming to

find the optimal solution in mesh simplification. The proposed area based mesh simplification uses a global search to find an optimal vertex and updates the vertices information at every iteration. Since the proposed algorithm tries to find the optimal vertex at every iteration, it is a greedy algorithm.

3.2 Related previous works

Mesh simplification methods can be categorized based on the kind of simplification function they employ [15, 16]. Heckbert [16] reviewed several different mesh simplification algorithms. Also Cignoni [17] investigated several simplification algorithms and gave a comparison between them. In this section, we classify previous simplification methods based on their distortion measures such as *Energy function, Volume, Distance, Quadric metric* and *Global error*.

3.2.1 Energy function based criterion

Using a simple local operator, Hoppe [4, 18, 19] introduced an energy function consisting of distance, representation and spring energy. The energy function directly measures the distortion between the final approximation and the original. For mesh simplification Hoppe used edge collapse, edge split and edge swap functions together and employed a mesh simplification technique based on the energy function to develop progressive mesh representation. A progressive mesh generates continuous sequence of level-of-detail approximations. Compared to other mesh simplification algorithms, Hoppe's algorithm requires more computations.

3.2.2 Volume based criterion

For distortion measure, Eastlick [20] introduced error polyhedra which is determined by the volume of a three simplex. Eastlick's algorithm is a combination of error polyhedra and dynamic triangulation algorithm. Eastlick also investigated a new neighborhood triangulation method which is based on the dynamic use of discrete differential operators. Eastlick's volume based error measure does not significantly increase the complexity needed for mesh simplification. Lindstrom [21] used edge collapse functions and volume of tetrahedra as a distortion measure. Alliez [22] introduced a mesh approximation algorithm using the volume between the simplified and the original mesh using a gradient-based optimized algorithm and edge collapse operator.

3.2.3 Distance based criterion

Mesh simplification methods based on distance are very simple. Schroeder [1] employed vertex distance to average plane or vertex distance to edge. In Schroeder's algorithm, all of the vertices having distances less than a predefined distance are deleted. If the n-gon is not on the boundary, the distance from a candidate vertex to an average plane is used. If the n-gon is a boundary polygon, the distance from the candidate vertex to edge is used.

3.2.4 Quadric metrics based criterion

Mesh simplification based on quadric error metrics (QEM) was introduced by Garland [10,23]. This mesh simplification algorithm is based on iterative edge contraction. To assign a priority to vertex for removal, they associate a symmetric matrix with each vertex. Then the distortion error at each vertex is defined using quadric form. Whenever edge contraction occurs and two vertices are merged to one vertex the new quadric metric is calculated using a simple additive rule. Since this mesh simplification method does not require recalculating the quadric metric, it is very fast. Velho [24] modified mesh structure using edge swap and a degree 4 vertex removal function. Cohen [25] proposed an algorithm generating a hierarchy of level-of-detail approximation for 3D objects. Cohen's algorithm guarantees that all of the points of the approximation are within user-specified distance from the original model. QEM were also adopted for surface simplification in [23] and [10].

3.3 Proposed mesh simplification algorithm

3.3.1 Area based error criterion

An area based error criterion considers the change of surface between original and approximation 3D objects. This criterion assumes the change of surface without rendering the final approximation. Since the triangle is the elementary unit of a polygon, for this criterion, the area of a triangle is required. Let $\psi(3) = \{v_i = (x_i, y_i, z_i) \in \mathbb{R}^3 | i = 1, 2, 3\}$ represent a triangle in a mesh. The area of the triangle is half the norm of the cross product of two vectors $(v_2 - v_1)$ and $(v_3 - v_1)$:

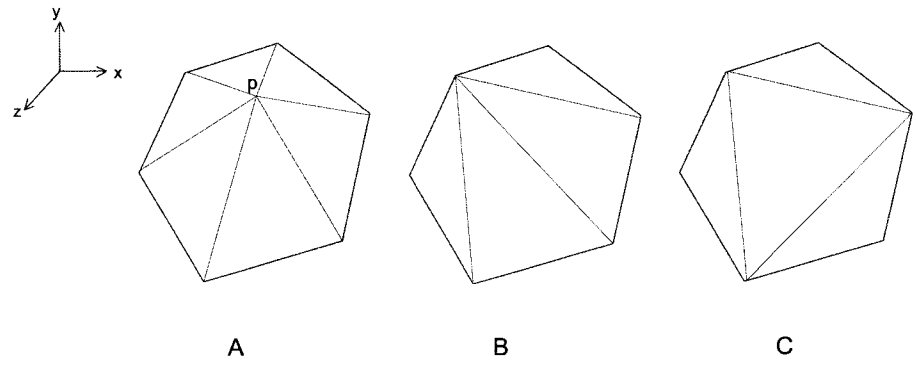


Figure 3.1: Mesh change from 6 faces to 4 faces after removing vertex p

$$\begin{aligned} \mathfrak{A}(\psi) &= \tfrac{1}{2}\|(v_2 - v_1) \times (v_3 - v_1)\| \\ &= \tfrac{1}{2}\|\alpha e_x + \beta e_y + \gamma e_z\| \\ &= \tfrac{1}{2}\sqrt{\alpha^2 + \beta^2 + \gamma^2} \end{aligned} \tag{3.1}$$

where $\alpha = (y_2 - y_1)(z_3 - z_2) - (z_2 - z_1)(y_3 - y_1)$, $\beta = (z_2 - z_1)(x_3 - x_1) - (x_2 - x_1)(z_3 - z_2)$ and $\gamma = (x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)$, and e_x, e_y and e_z are unit vector of x, y and z direction, respectively.

Using the area of triangle $\mathfrak{A}(\psi)$ we define an area based distortion between two meshes. For example, in Figure 3.1, if the shape of the mesh is changed from the mesh of Figure 3.1(a) to the mesh of Figure 3.1(b) by removing vertex p, the distortion will be the absolute value of the difference between the total area of the mesh of Figure 3.1(a) and the total area of the mesh of Figure 3.1(b). We express the meshes of Figure 3.1(a) and 3.1(b) with topological function as

$\Psi_A = \{\psi_{A_1}, \psi_{A_2}, \cdots, \psi_{A_6}\}$ and $\Psi_B = \{\bar{\psi}_{B_1}, \bar{\psi}_{B_2}, \cdots, \bar{\psi}_{B_4}\}$. Then the area based distortion of vertex p is given by

$$\begin{aligned} \mathfrak{D}_{area}(p) &= |\mathfrak{A}(\Psi_A) - \mathfrak{A}(\Psi_B)| \\ &= \left| \sum_{i=1}^{6} \mathfrak{A}(\psi_{A_i}) - \sum_{i=1}^{4} \mathfrak{A}(\bar{\psi}_{B_i}) \right|. \end{aligned} \tag{3.2}$$

If an object S_1 is simplified to an approximation S_2 using the area based distortion by removing n vertices, the total distortion between S_1 and S_2 is

$$\begin{aligned} \mathfrak{E}_{area}(S_1, S_2) &= \int_{\psi \in S_1} \mathfrak{A}(\psi) dS_1 - \int_{\psi \in S_2} \mathfrak{A}(\psi) dS_2 \\ &= \sum_{i=1}^{n} \mathfrak{D}_{area}(v_i) \\ &= \sum_{i=1}^{n} \left| \sum_{k=1}^{\rho(v_i)} \mathfrak{A}(\psi_{ik}) - \sum_{k=1}^{\rho(v_i)-2} \mathfrak{A}(\bar{\psi}_{ik}) \right| \end{aligned} \tag{3.3}$$

where ψ_{ik} and $\bar{\psi}_{ik}$ is the kth face related to vertex v_i of an object and its approximation respectively.

3.3.2 Candidate vertices

All vertices except boundary vertices (as defined in Section 2) and vertices which are center vertex to a non-convex polygon are initial candidate vertices for removal in proposed mesh simplification algorithm. Figure 3.2 shows the example of the distortion when boundary vertex is considered as candidate vertex. Figure 3.2(b) and 3.2(c) are the experimental results when the algorithm consider boundary vertex as non candidate vertex and candidate vertex, respectively. Although Figure 3.2(c) represent one more face the distortion is more serious than the representation of Figure 3.2(b). In the initialization stage, for every candidate vertex we collect the information about coincidence faces and the degree of that vertex. Figure

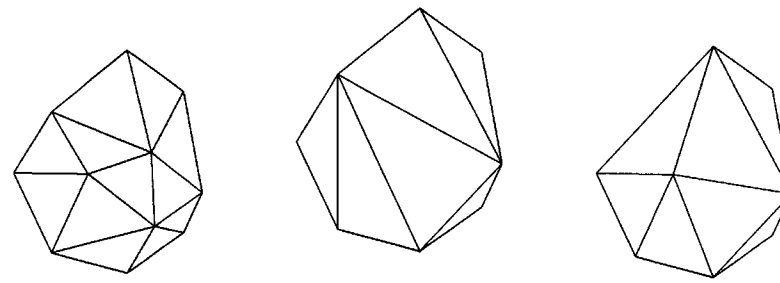


Figure 3.2: Polygon representation: (a) original polygon, (b) after simplification without considering boundary vertex as candidate vertex, (c) after simplification with considering boundary vertex as candidate vertex.

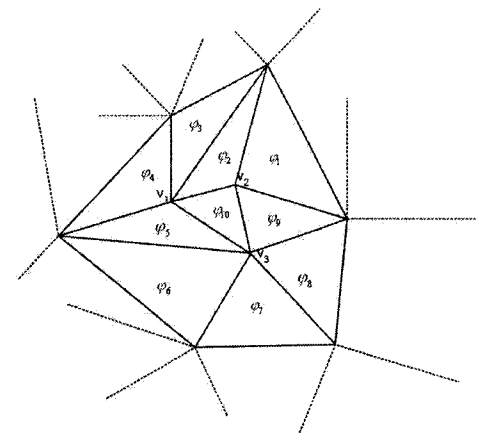


Figure 3.3: Coincidence face and degree of each candidate vertex.

3.3 shows an example of the initialization. Vertex v_1 cannot be a candidate vertex because it is the center vertex to a non-convex polygon. v_2 and v_3 are valid candidate vertices. The degree of each vertex is $\rho(v_2) = 4$ and $\rho(v_3) = 6$, respectively. The coincidence faces of vertex v_2 are ψ_1, ψ_2, ψ_9 and ψ_{10}. The coincidence faces of vertex v_3 are $\psi_5, \psi_6, \psi_7, \psi_8, \psi_9$ and ψ_{10}.

When a candidate vertex is removed from a mesh, the mesh can be re-triangulated in different ways with different distortion. For example, Figures 3.1(b) and 3.1(c) show two different re-triangulations of mesh of Figure 3.1(a) after removing vertex p. The proposed algorithm, using area based distortion, finds the best re-triangulation type for each candidate vertex. The distortion between the original and re-triangulated mesh is stored to be compared with that of other candidate vertices. We will describe the re-triangulation procedure in Section 3.3.3. Corresponding to each candidate vertex there are 6 information: (1) the degree of the vertex, (2) re-triangulation type, (3) the area of polygon ($\mathfrak{A}(\Psi)$), (4) the area of remodel type, (5) the distortion which is difference between the area of polygon and the area of remodel type and (6) the index number of the coincidence faces of the vertex. After sorted by distortion, all of the candidate vertices are stored in a priority heap. At every iteration the candidate vertex located on the top of the priority list is removed.

3.3.3 Simplification algorithm

The structure of a mesh consists of different kinds of vertices in terms of degree. In our algorithm, all vertices with a degree greater than 3 which are not center vertex to a non-convex polygon and are not a boundary vertex are considered as a

candidate vertex. Let consider the mesh of Figure 3.1(a) and vertex p as a candidate vertex. In Figure 3.1(a) the mesh consists of six triangles, whereas mesh of Figure 3.1(b) consists of four triangles. Since we reduce the number of vertices by deleting vertex point p, in terms of data, the mesh of Figure 3.1(b) require less data than the mesh of Figure 3.1(a). In every iteration, proposed mesh simplification algorithm removes 2 triangles and 1 vertex. In rendering the 3D mesh, the structure of the mesh of Figure 3.1(b) is simpler than that of the mesh of Figure 3.1(a).

The proposed mesh simplification algorithm consists of three processes: (1) find an optimal vertex, (2) find a re-triangulation type and (3) update all vertices of the simplified polygon. To find an optimal vertex it is required to know the re-triangulation type of the polygon.

Algorithm 1 Mesh Simplification

1 Read geometrical and topological data of object $S(\mathcal{V}, \Psi)$.
2 Initialize mesh structure:
(1) Calculate the area of each face ($\mathfrak{A}(\psi)$).
(2) Build candidate vertices of object.
(3) Choose the re-triangulation type of each candidate vertex (use *Optimum re-triangulate*).
(4) Build priority heap of candidate vertex based on $\mathfrak{D}_{area}(v)$.
repeat
3 Remove the vertex located on top priority heap.
(1)Delete all of the coincidence faces of optimal vertex from topological data set.
(2)Delete vertex from geometrical data set.
(3)Re-triangulate the polygon and add new faces to topological data set.
3 Update the information of candidate neighbor vertices.
4 Update the re-triangulation type of each candidate neighbor vertex (use *Optimum re-triangulate*).
5 Update priority heap.
until The number of faces is less than predefined the number of faces.

Finding an optimum vertex

The proposed mesh simplification algorithm is represented in Algorithm 1. In the initialization step, the distortion of all candidate vertices are compared to each other to form a priority heap. The distortion of different re-triangulations are calculated and compared in *optimum re-triangulate* function. Then, the vertex which has minimum distortion is removed. The polygon corresponding to the removed vertex is re-triangulated with a fewer number of triangles. Then the priority heap and the information for candidate neighbor vertices are updated. Thus, this algorithm at each step chooses the vertex having minimum distortion. In the next subsection we describe how our algorithm chooses the triangulation having minimum distortion.

Choose optimum re-triangulation

Algorithm 2 Optimum re-triangulate

1 Accept a candidate vertex v.
2 Form a polygon using coincidence triangle of vertex v (Ψ_v).
2 Build n different re-triangulation types (Ψ_i $i = 1, 2, 3, \cdots n$).
3 Choose one re-triangulation type having minimum distortion $\mathfrak{D}_{area}(v) = \min_i |\mathfrak{A}(\Psi_v) - \mathfrak{A}(\Psi_i)|$.
4 Return re-triangulation type and distortion to main function.

The *optimum re-triangulation* function is described in Algorithm 2. The distortion of different re-triangulations are compared to each other. The function returns the optimal re-triangulation and the distortion to the main function. There is a trade-off between the distortion and complexity. Considering more re-triangulation options guarantees less distortion, but there is more complexity to compare the distortion.

Table 3.1: The number of re-shape model

degree of vertex	4	5	6	7	···
No. of re-shape model	2	5	9	9	9

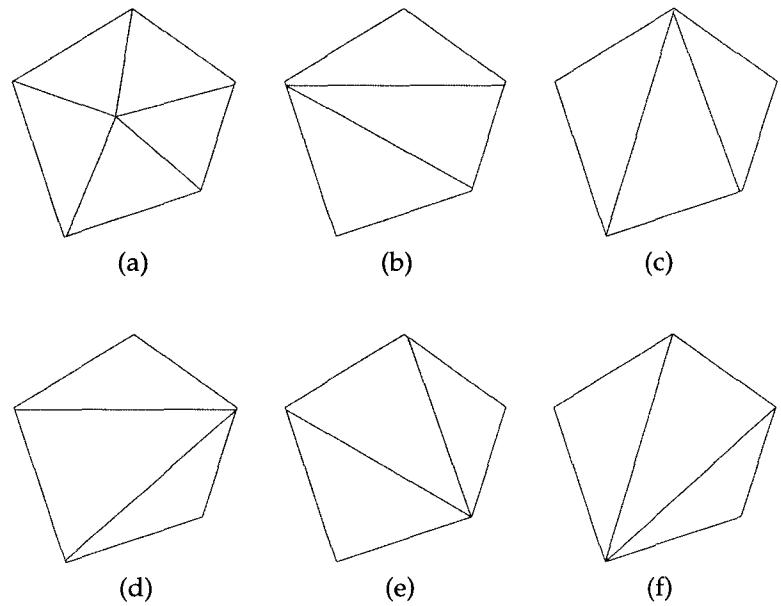


Figure 3.4: Reshape model of pentagon: (a) original mesh structure, (b)-(f) candidate reshape model

Table 3.1 shows the number of re-triangulations as a function of the degree of the candidate vertex. As the degree of center vertex increases, the number of re-triangulations increases. To reduce computational complexity, we consider only 9 re-triangulations if the degree of the center vertex is more than 6. Figure 3.4 illustrates a pentagon shaped mesh structure and five possible re-triangulations. Remember that all of the re-triangulations have the same number of triangles and the same number of vertices. Thus, in terms of data rate, all of them are the same.

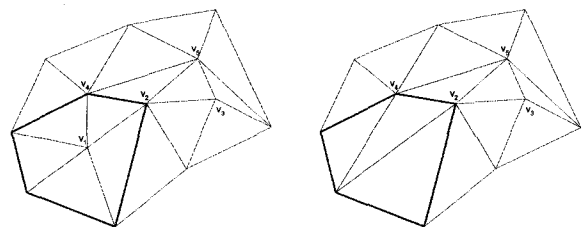


Figure 3.5: Mesh simplification with candidate vertex v_1 and related candidate vertices must be updated (v_2, v_4): (a) before simplification and (b) after simplification

Update information of neighboring vertices

Whenever we remove a vertex, the information of its neighboring vertices changes. There are five candidate vertices in Figure 3.5(a) ($v_1, \cdots v_5$). At the initial stage, the degree of vertices are: $\rho(v_1) = 5, \rho(v_2) = 6, \rho(v_3) = 4, \rho(v_4) = 6$ and $\rho(v_5) = 6$. If we assume that vertex v_1 is removed, then the polygonal area marked by thick edges can be re-triangulated as Figure 3.5(b). Although the degree information of v_2 and v_4 are the same, the area of polygons $\mathfrak{A}(\Psi_{v_2})$ and $\mathfrak{A}(\Psi_{v_4})$ have changed (Ψ_{v_2} and Ψ_{v_4} represent the polygon with v_2 and v_4 as center vertex, respectively). Thus area based distortion $\mathfrak{D}_{area}(v_2)$ and $\mathfrak{D}_{area}(v_4)$ need to be updated.

Whenever the information of a vertex is changed, the calculation of distortion and resorting of the priority list is required. This increases the computational complexity of the proposed algorithm. In order to reduce the computational load, we first check to see whether a neighboring vertex needs to be updated. Figure 3.6 is an example of updating the information of neighbor vertices. Consider, in Figure 3.6(a), vertex v_1 is the optimal vertex. All of the coincidence triangles ($\psi_7, \psi_8, \psi_9, \psi_{10}, \psi_{11}, \psi_{12}, \psi_{13}$) are removed and re-triangulated as Figure

3.6(b) with $\bar{\psi}_7, \bar{\psi}_8, \bar{\psi}_9, \bar{\psi}_{10}$ and $\bar{\psi}_{11}$. The neighbor vertices of the optimal vertex v_1 are $v_2, v_3, v_4, v_5, v_6, v_7$ and v_8. As mentioned before, if we want to keep the shape of the object the boundary vertex must not be removed. Thus, only the information for vertex v_7 and v_8 are updated. In this case, we need to update two surface area Ψ_{v_7} and Ψ_{v_8} as

$$\begin{aligned} \mathfrak{A}(\Psi_{v_7}) = \; & \mathfrak{A}(\Psi_{v_7}) \\ & -(\mathfrak{A}(\psi_{12}) + \mathfrak{A}(\psi_{13})) \\ & +\mathfrak{A}(\bar{\psi}_{10}) \end{aligned} \tag{3.4}$$

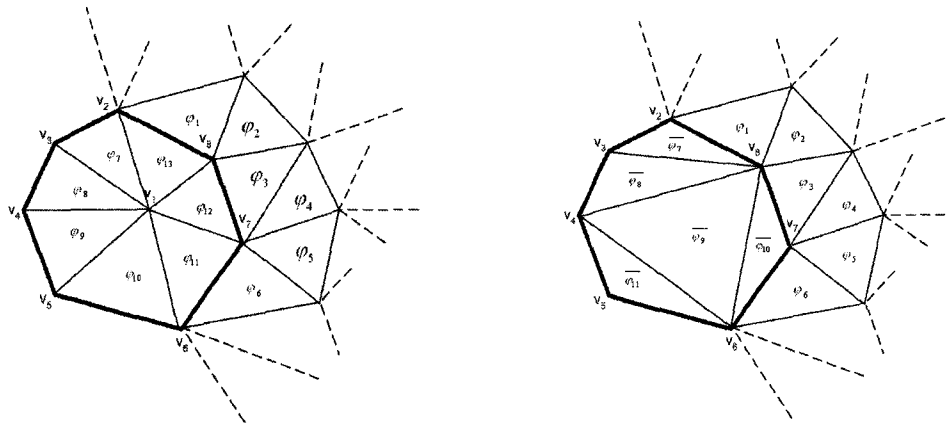
and

$$\begin{aligned} \mathfrak{A}(\Psi_{v_8}) = \; & \mathfrak{A}(\Psi_{v_8}) \\ & -(\mathfrak{A}(\psi_7) + \mathfrak{A}(\psi_{13})) \\ & +(\mathfrak{A}(\bar{\psi}_7) + \mathfrak{A}(\bar{\psi}_8) + \mathfrak{A}(\bar{\psi}_9) + \mathfrak{A}(\bar{\psi}_{10}) \end{aligned} \tag{3.5}$$

After updating the information of those two vertices (v_7 and v_8), we resort the priority list.

3.4 Experimental results

We use four 3D objects to verify the performance of the proposed mesh simplification algorithm. The 3D objects are Face, Mountain, Venus and Head as shown in Figures 3.7-3.10. For each model, we reduced the number of faces to between 90% and 20% of the number of faces of original model. Table 3.2 summarizes the number of faces of original model and approximation.



(a) (b)

Figure 3.6: Update the information of neighbor vertex by removing optimal vertex v_1: (a) before simplification and (b) after simplification

Table 3.2: The number of face and vertex of test models.

	Face		Mountain		Venus		Head	
	face	vertex	face	vertex	face	vertex	face	vertex
100%	826	512	4,802	2,500	1,396	711	1,252	654
90%	742	441	4,320	2,244	1,256	641	1,127	591
50%	412	260	2,400	1,267	698	361	626	337
20%	164	119	959	521	279	150	250	144

Figure 3.7 shows the original and simplified Face object. Figure 3.7(a) is the original having 826 faces. Figure 3.7(b) is the approximation with 90% (742 faces) of the original. It is difficult to see the difference between the Figures 3.7(a) and 3.7(b). Figure 3.7(c) is the approximation with 50% (412 faces) of original data. When we approximate with 50%, we can see the distortion between the original and the approximation. The distortions at the forehead and nose area are most obvious. There is more distortion in the approximation with 20% as we can see in Figure 3.7(d). The detail of the face is almost gone. From Figures 3.7(a)-3.7(d) we can realize that using our method preserves the shape of Face. In particular, it is worth noting that the holes for eyes are not changed. Since all of the vertices on the hole are boundary vertex, those vertices are not removed. Also all of the vertices on the boundary of the Face are left intact. Thus, holes and the shape are preserved.

Figure 3.8 shows the experimental results with the Mountain object. The original has 4,802 faces as shown in Figure 3.8(a). Figures 3.8(b)-3.8(d) are approximations with 90 , 50 and 20% of the original data, respectively. When we reduce the number of faces to around 50%, visible distortion starts to occur. Although, in the 20% approximation, we have lost the detail of the center part of the Mountain the whole shape is preserved rather well and we can easily perceive this object as a mountain.

Figure 3.9 is the result of mesh simplification with the Head model. The original consists of 1,252 faces. There are two parts in the Head model: the face area and the neck area. As we can see from Figures 3.9(a) and 3.9(b), Figure 3.9(b) looks similar to the original. In Figure 3.9(c) the distortion occur on the forehead and

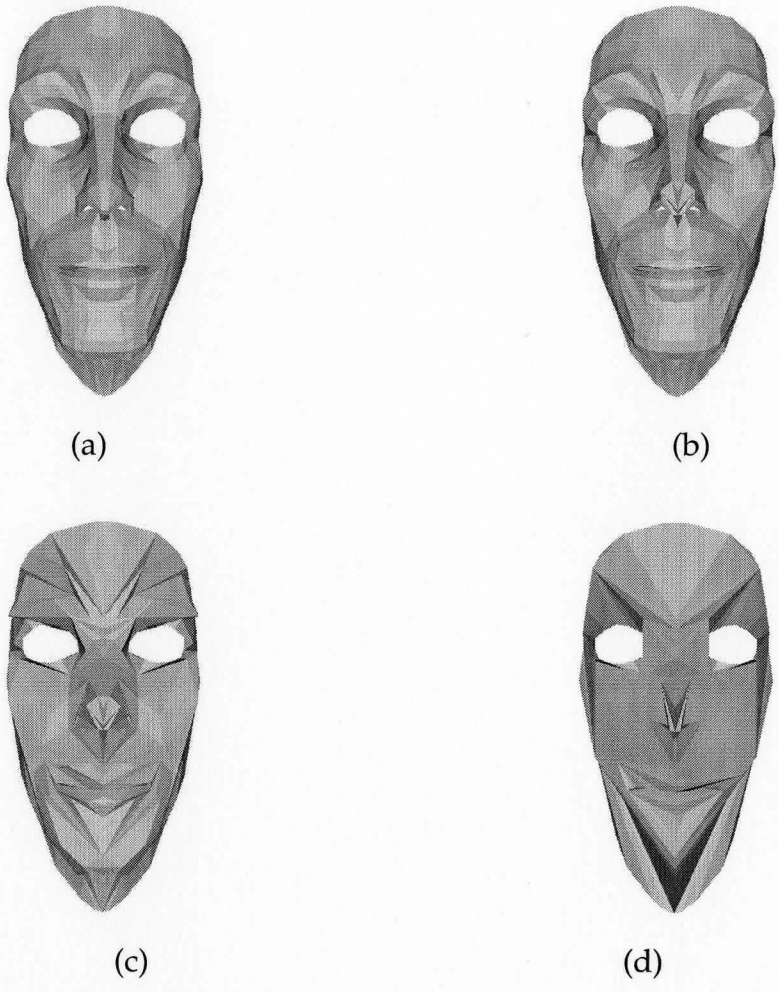
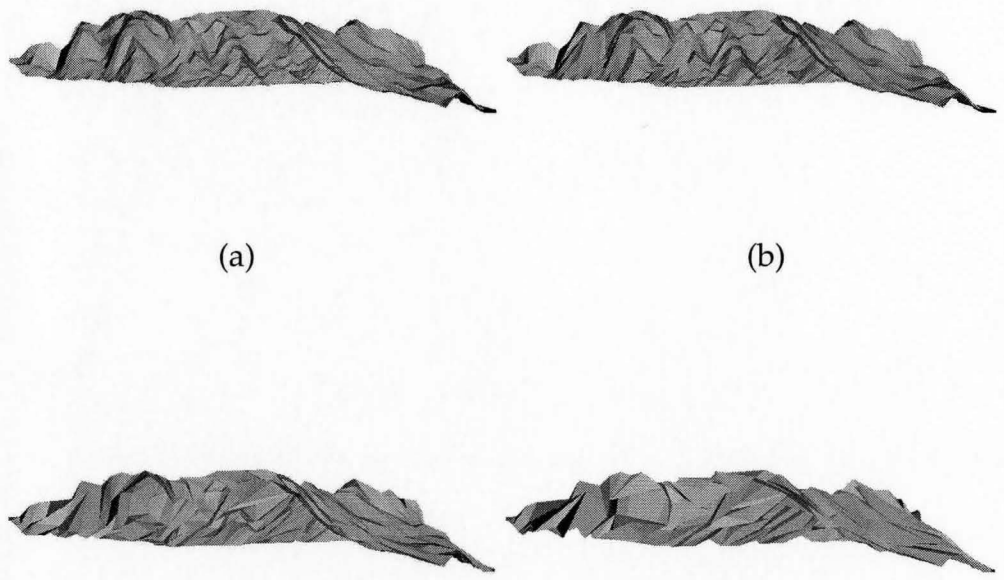


Figure 3.7: 3D representation of Face: (a) original model with 826 faces, (b) approximation with 742 faces, (c) approximation with 412 faces and (d) approximation with 194 faces.



(a) (b)

(c) (d)

Figure 3.8: 3D representation of Mountain: (a) original Mountain with 4802 faces, (b) approximation with 4320 faces, (c) approximation with 2400 faces and (d) approximation with 959 faces.

mouth area and the nostril is removed. The vertices on the hole of the nose are not boundary vertices. Thus, those vertices are removed. Since most of vertices on the ear are boundary vertices, the ears are well preserved. When the original Head model is approximated with around 20% of original data, most of the detail of the face is removed. However, we can still distinguish the face area from the neck area.

Figure 3.10 shows the result with a simple object, Venus. The original Venus has 1,396 faces. As we can see from Figure 3.10(a), in the Venus, there are no holes and it consists of one large object. Figures 3.10(b)-3.10(d) show its approximation with 90, 50 and 20% of the number of faces, respectively. Although we remove the number of faces to around 20% of original object, similar to other objects, the shape is well preserved. The boundary of two legs remain intact.

To verify the quality of the approximation of the proposed mesh simplification algorithm, we compare the results of our algorithm to the results of mesh simplification based on quadric metric which was proposed by Garland [10]. In terms of speed, proposed algorithm requires around 1.5 times of processing time of quadric metric based mesh simplification algorithm. To quantitatively compare the quality of the approximations of two algorithms, we employed *Metro* [9]. We compared the approximation results of the proposed and quadric metric based mesh simplification algorithms, at 90, 50 and 20% using $\mathfrak{E}_{max}$, $\mathfrak{E}_{avg}$ and $\mathfrak{E}_{rms}$. In Figures 3.11 - 3.13 QSlim based method represent distortion graphs of approximation using quadric metric based mesh simplification algorithm [23].

Figures 3.11(a)-3.11(d) are the error curves using the Hausdorff distance measure $\mathfrak{E}_{max}$ for Face, Mountain, Head and Venus model, respectively. In all of the

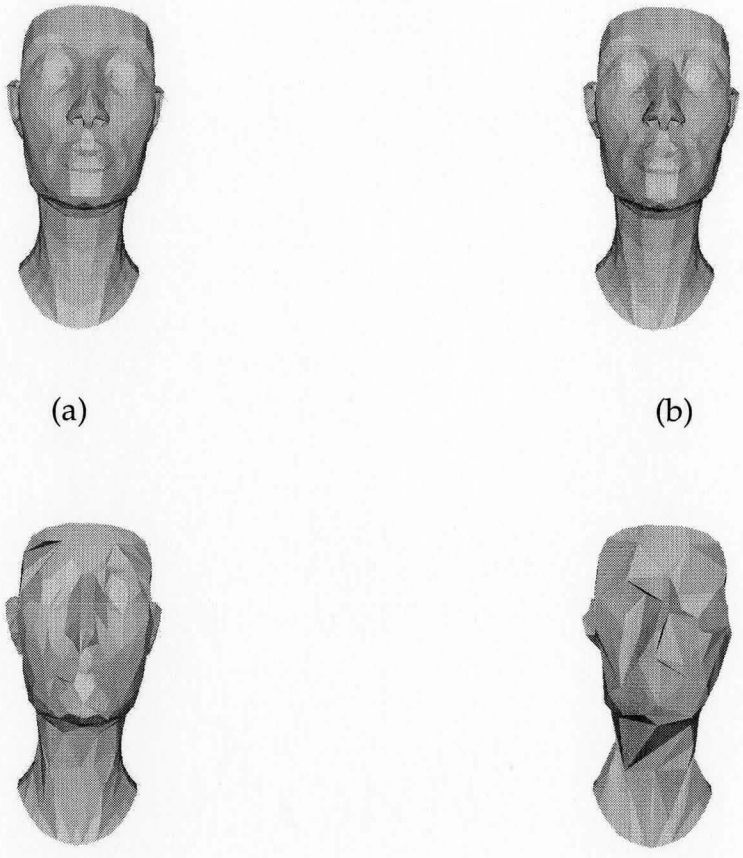


Figure 3.9: Original 3D representation of head: (a) original Head with 1252 faces, (b) approximation with 1127 faces, (c) approximation with 626 faces and (d) approximation with 250 faces.

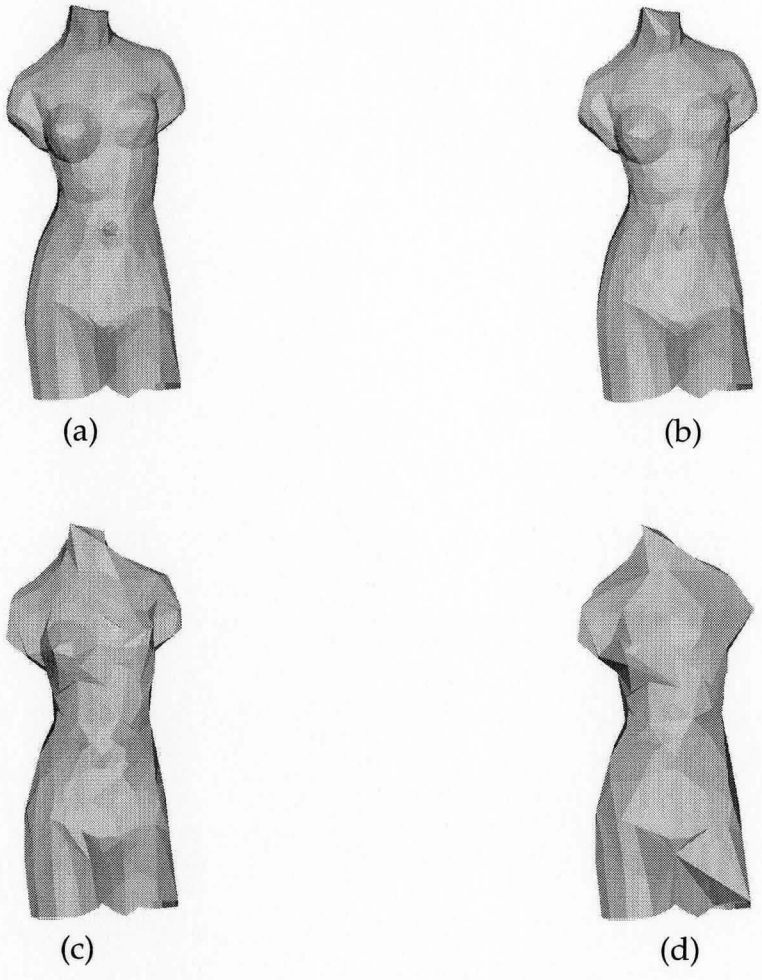


Figure 3.10: Original 3D representation of Venus: (a) original Venus with 1396 faces, (b) approximation with 1256 faces, (c) approximation with 698 faces and (d) approximation with 279 faces.

models, the results of the proposed mesh simplification algorithm are better than those of the quadric metric based mesh simplification algorithm. In the case of the Mountain model, we can see that the proposed algorithm generates much better approximations. The common property of our proposed algorithm is that when the number of faces is small the distortion increases rapidly. This is because the proposed algorithm tries to keep the boundary vertices and remove only vertices of non-boundary area.

Figure 3.12 shows the simulation result of error curves using the average distance $\mathfrak{E}_{avg}$ between the approximation and the original. Figures 3.12(a)-3.12(d) represent the results of Face, Mountain, Head and Venus models, respectively. In the case of the Face model, as we can see from Figure 3.12(a), the proposed algorithm has better performance than mesh simplification algorithm based on quadric metric until around 50% of original data. The property of error curve using average distance of proposed algorithm is that the error increases substantially when we reduce the data rate to less than 40% of the original data. But, the result is still comparable to the quadric metric based mesh simplification algorithm.

Figure 3.13 shows another simulation result error curves of $\mathfrak{E}_{rms}$. Figures 3.13(a)-3.13(d) are results of Face, Mountain, Head and Venus model, respectively. Also Figure 3.13 shows that the proposed mesh simplification algorithm outperforms the mesh simplification algorithm based on quadric metric. In the case of the Head model, the proposed algorithm generates much better results than the quadric metric based simplification algorithm. When there are holes within the objects, for example the Face object, the proposed mesh simplification algorithm introduces much less distortion than the quadric metric based mesh simplification algorithm.

While the quadric metric based mesh simplification algorithm removes the eye holes of the Face object, the proposed algorithm keeps these holes. The proposed algorithm maintains holes because their boundary vertices are never removed thus preserving topological features of the objects.

3.5 Conclusions

In this chapter, we have described a mesh simplification using area based distortion. The algorithm reduces the number of faces required for representation of a polygon in the 3D object. The optimal vertex is chosen by an area-based distortion criterion. To minimize the distortion between the original and final approximation, various re-triangulations are considered. A priority heap is used to reduce computational complexity. The proposed mesh simplification algorithm reduces data rate without introducing serious deformation. Especially for the case when there are holes in the object, the proposed algorithm approximates the original object with small distortion. Since our mesh simplification algorithm does not remove boundary vertex, the shape of the object is retained.

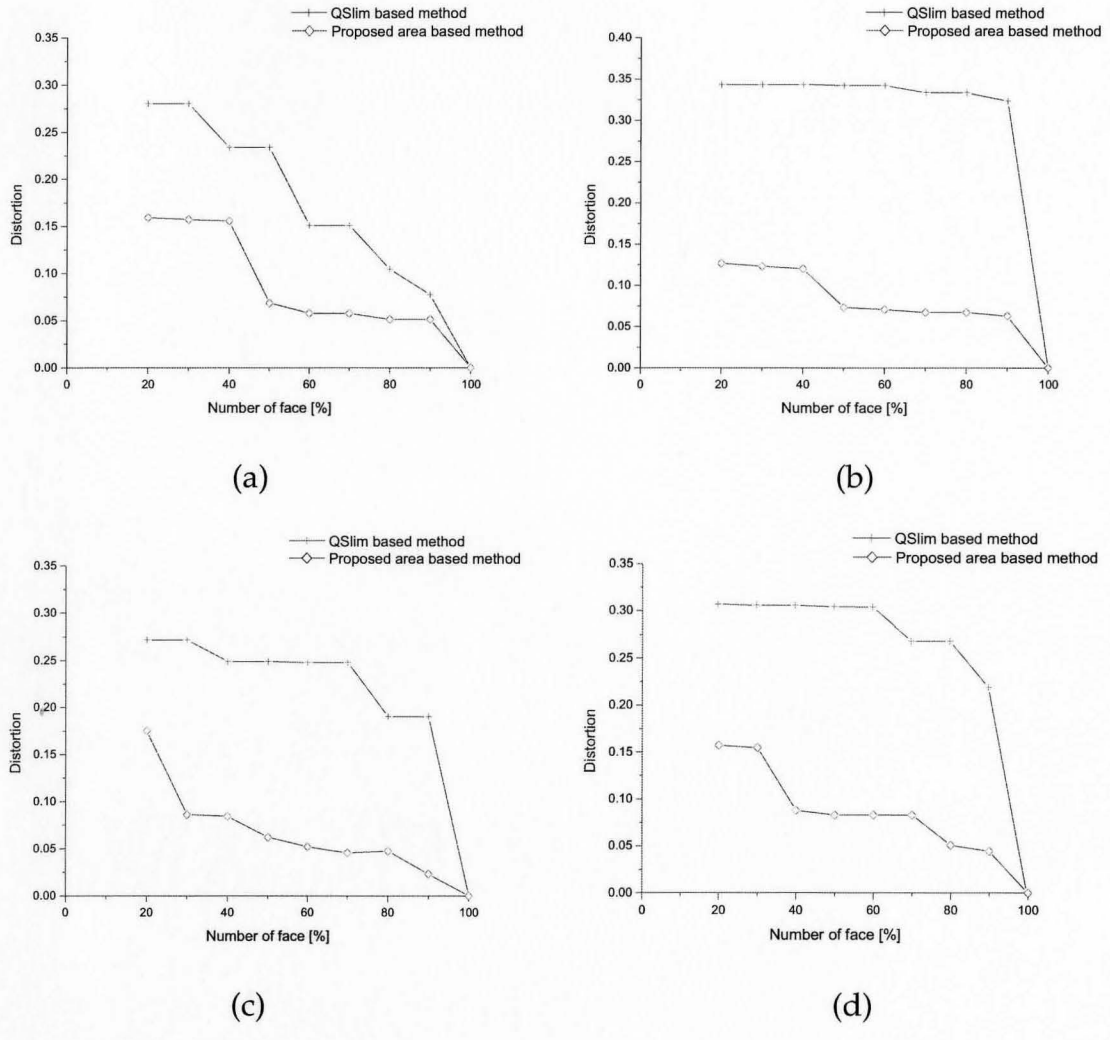


Figure 3.11: Error curve based on Hausdorff distance ($\mathfrak{E}_{max}$): (a) Face, (b) Mountain, (c) Head and (d) Venus

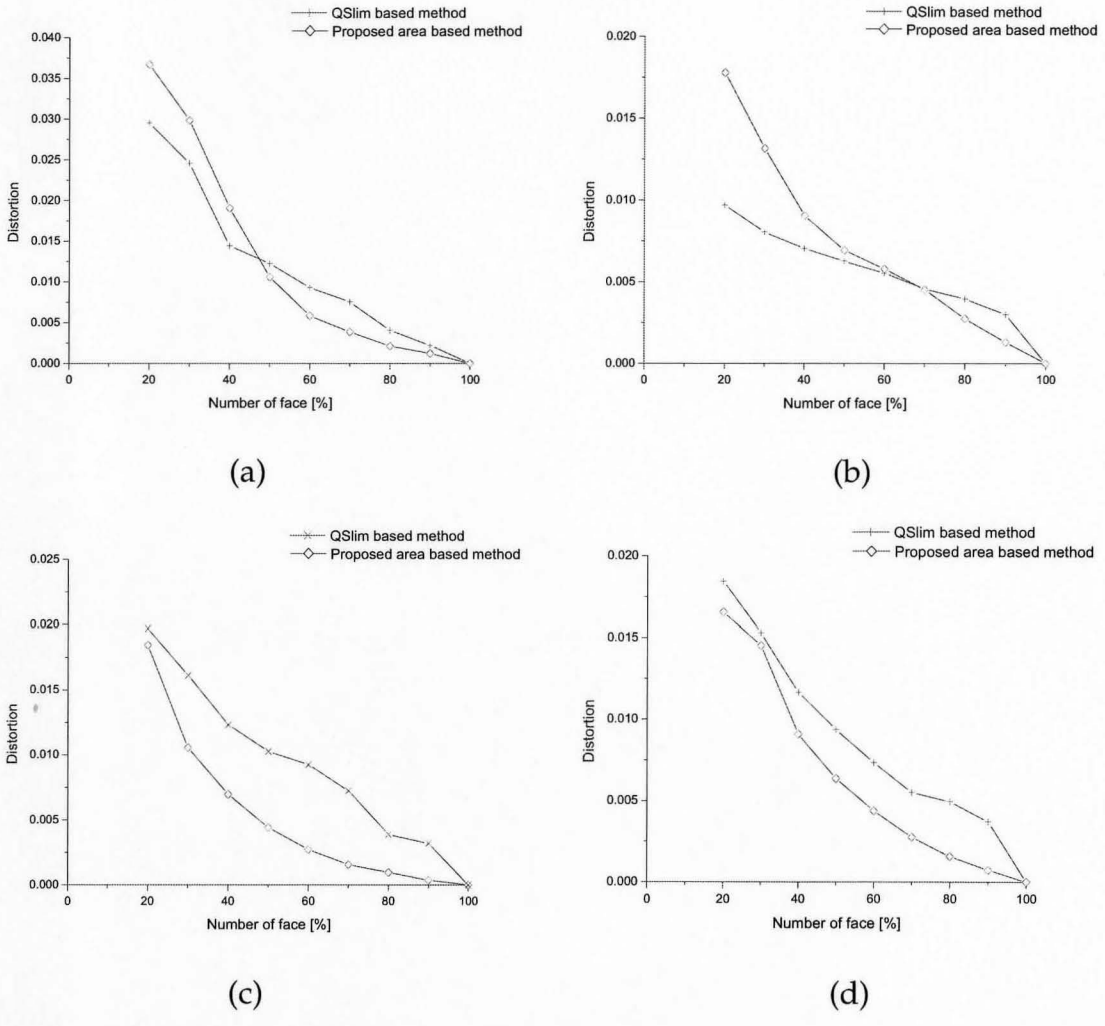


Figure 3.12: Error curve based on average distance ($\mathfrak{E}_{avg}$): (a) Face, (b) Mountain, (c) Head and (d) Venus

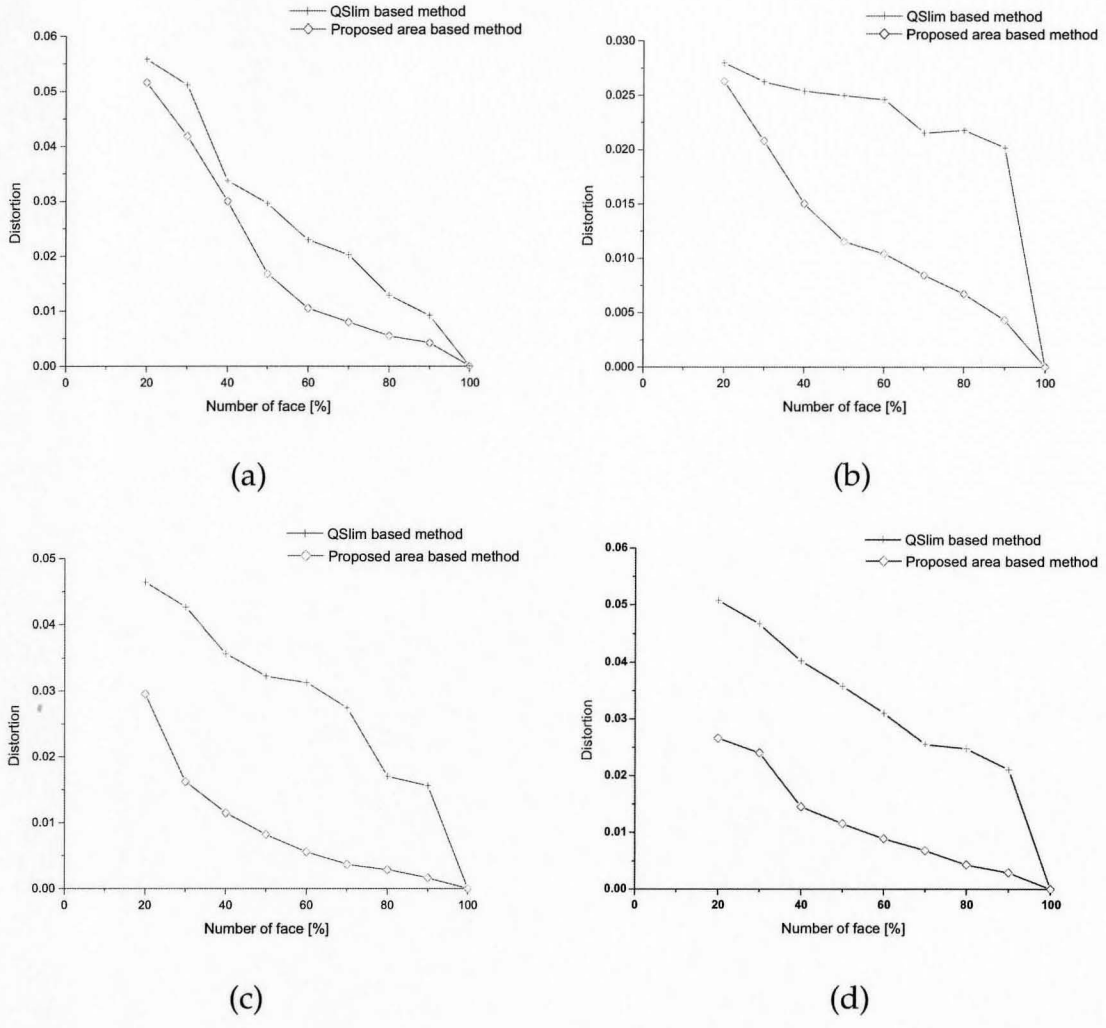


Figure 3.13: Error curve based on root mean square distance ($\mathfrak{E}_{\mathrm{rms}}$): (a) Face, (b) Mountain, (c) Head and (d) Venus

Chapter 4

Communication of 3D graphic models

4.1 Introduction

In the era of Internet there are a lot of chances of sending graphic data over unreliable network. In this condition error resilient techniques for data transmission should be considered. Depending on the network condition, different user require different quality of service. In a diversity-based network, encoder transmits data to the decoder through different channel [26]. In this environment, although one or more channel breaks down, the decoder can get enough data to reproduce the input source.

The simplest method using diversity-based channel is sending same data through each channel. Although when one of the channels is broken this method work, but it is redundant if all channels are functional. Multiple description coding (MDC) is an alternative. MDC generates different descriptions which are correlated with

each other. MDC describes a source with two (or more) descriptions to be transmitted via diversity based communication channels [26]. After Wolf *et al.* [27] introduce MDC a lot of progress in designing MDC have been done [28]. In MDC the description transmitted over each channel is enough to reproduce original object at the decoder with a guaranteed minimum distance between original and reproduced source. If there is no channel error the decoder reproduces a higher quality of object than reproduced from each channel. In this thesis we only consider two channels and two descriptions.

The goal of MDC is generating two descriptions for an input source, such that the difference between the original and the reconstructed source is minimized with the condition that the difference should be less than predefined thresholds in worst case (just one description is available) [29]. In this thesis we concentrate on MDC based on scalar or vector quantization. The performance of MDC depends on two main procedures: quantization and index assignment (IA). Designing optimal quantizer relies on the distribution of the source and channel capacity [30].

IA is directly related to the bit-rate and the distortion when all descriptions are not available at the decoder. Thus the designing of IA is the most important part in the theory and practice of MDC [31]. Designing a pair of multiple description encoder and decoder is based on the assumption of on-off channel condition: the channel works perfectly or is broken. Multiple description scalar quantizer (MDSQ) was proposed by Vaishampayan [32]. Vaishampayan's IA for scalar quantization was proved as optimal IA in terms of mean squared error (MSE) criterion [32].

In decoding of the descriptions, the performance of MDC relies on the level of

quantization if both descriptions are available at the decoder. When one of channels is broken the performance of MDC depends largely on the decoding method rather than the level of quantization itself. This is the issue of designing side decoder. One method to approximate the lost description is assuming that IA matrix is diagonal. Using diagonal IA matrix encoder generates exactly same two descriptions. Since usually diagonal IA matrix is not used at the encoder difference between original and reproduced data exists. The other method is using probability of input source. This method requires sending the probability of quantization index number as side information. The size of side information is same with the codebook size of quantization.

4.2 Multiple description coding

Multiple description encoder generates a pair of redundant descriptions to represent one source input. Multiple description consists of two main procedures which are quantization ($q(.)$) and index assignment ($a(.)$). While quantization is applied to reduce the bit-rate of input source index assignment is adopted to represent the index number of quantization efficiently. Let assume a sequence of input source $X = \{x_1, x_2, \cdots, x_n\}$. In quantization, we assume there is a quantizer mapping the input random variable $X \in \mathbb{R}^n$ to the index of the codebook $J_l = \{1, 2, \cdots, l\}$ where l is the codebook size. Although different quantization methods should be used for different distribution of input source, at higher rates, uniform quantizer is the optimal quantizer [33]. In case of uniform scalar quantizer the quantization step size $\Delta = (x_{max} - x_{min})/l$, where x_{max} and x_{min} are maximum and minimum value of input source X, respectively. By inverse quantization there are l dequantized

values which are represented as $\hat{x}_j = x_{min} + (j - 0.5) \times \Delta, j = 1, 2, \cdots, l$. Then the quantization procedure is formulated as

$$q(x) = \arg\min_j \|x - \hat{x}_j\|^2 \quad \forall j \in l.$$

Dequantization procedure is simply the inverse of quantization which is $q^{-1}(j) = \hat{x}_j$.

In IA the function $a(.)$ maps one index in the quantization codebook into a pair of indices (descriptions). The IA can be represented as $a : J_l \rightarrow J_m \times J_m$ where $J_m = \{1, 2, \cdots, m\}$ and m^2 is the size of IA matrix. Figure 4.1 shows two modified nested IA matrices with different off diagonal which are designed by Vaishampayan [32]. When IA matrix has k-off diagonal the number of central cells is given by $l = (2k + 1)m - k(k + 1)$. Thus, by increasing k, in the same size of IA matrix, we can increase quantization levels. When k has maximum value the number of quantization levels is simply $l = m^2$. Remember that the maximum k should be less than m. The IA procedure is more clear by representing $a(j) = (r, c)$ where j, r and c are indices satisfying $j \in J_l$ and $\{r, c\} \in J_m$, respectively.

The MD encoder can be defined as a sequence of quantization and IA $g(.) = a(q(.))$. In the encoder one index pair is generated for each source input as $g(x) = (r, c)$. Then index r is transmitted over channel 1 at rate R_1 and index c is transmitted over channel 2 at rate R_2. The total bit-rate required to send the information of index j is $R = R_1 + R_2$. The decoder is simply inverse procedure of encoder which is $f(.) = q^{-1}(a^{-1}(.))$. The inverse IA at the decoder depends on the channel condition

and represented as

$$\begin{aligned} j_0 &= a^{-1}(r,c) && \text{If both channels work} \\ j_1 &= a^{-1}(r) && \text{If only channel 1 works} \\ j_2 &= a^{-1}(c) && \text{If only channel 2 works.} \end{aligned} \tag{4.1}$$

At the decoder the regenerated data for each input source is

$$\hat{x}_i = q^{-1}(j_i) \quad i = 0,1,2 \tag{4.2}$$

and regenerated output for input source X is

$$\hat{X}_i = \{\hat{x}_{i1}, \hat{x}_{i2}, \cdots \hat{x}_{in}\} \quad i = 0,1,2 \tag{4.3}$$

The difference between input source x and regenerated output $\hat{x}$ at the decoder is $d(x, f(g(x)))$ where $d()$ is any distortion measure. Let mean-squared error of two vectors $X \in \mathbb{R}^n$ and $\hat{X} \in \mathbb{R}^n$ as distortion measure. Then $E[d(X,\hat{X})] = mse(X,\hat{X}) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2$. Then we call $d(X,\hat{X}_0)$ as central distortion, and $d(X,\hat{X}_1)$ and $d(X,\hat{X}_2)$ as side distortions. The goal of multiple description is designing the quantizer and IA matrix of encoder-decoder pair to minimize central distortion $d(X,\hat{X}_0)$ when both channels work, subject to constrains on two side distortions $d(X,\hat{X}_1)$ and $d(X,\hat{X}_2)$ are less than predefined distortion.

	1	2	3	4	5	6	7	8	md 1
1	1	3							
2	2	4	5						
3		6	7	9					
4			8	10	11				
5				12	13	15			
6					14	16	17		
7						⋱	⋱	⋱	
8							⋱	⋱	

md 2

(a)

	1	2	3	4	5	6	7	8	md 1
1	1	3	5						
2	2	6	8	10					
3	4	7	11	12	14				
4		9	13	16	17	19			
5			15	18	21	23	25		
6				⋱	⋱	⋱	⋱	⋱	
7					⋱	⋱	⋱	⋱	
8						⋱	⋱	⋱	

md 2

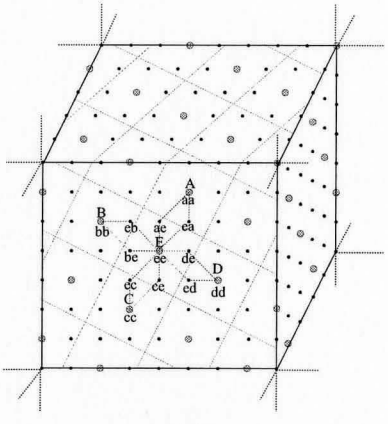
(b)

Figure 4.1: Example of IA: (a) matrix with one-off diagonal ($k = 1$) and (b) matrix with two-off diagonal ($k = 2$).

	1	2	3	4	5	6	7	8	md_1
1	1	2							
2	3	4	6						
3		5	7	8					
4			9	10	11				
5					12	14			
6					13	15	16		
7						17	18	20	
8							19	21	

md_2

(a)



(b)

Figure 4.2: Example of index assignment: (a) matrix of MDSQ for $m = 8$ and $k = 1$ and (b) mapping of MDVQ.

4.3 Multiple description for 3D graphics

The applications of MDC can be found in [34], [35], [36] and [37]for image coding. In [38] and [39] MDCs are used for video compression to transmit data over error prone channel. In this section and following subsection we deal with 3D graphic data as input source of MDC. At first we briefly study the structure of 3D graphics. Then scalar and vector quantizer for 3D graphic data are investigated.

4.3.1 Structure of 3D graphic and MDC

In 3D graphic representation, 3D data consists of geometrical and topological data. Geometrical data describe the location of each vertex of 3D graphic. Topological data is a set of vertex members of every polygon (the set of polygons forming the object). Thus, the structure of 3D graphic ($\mathcal{S}$) can be expressed as $\mathcal{S} = (\mathcal{V}, \Psi)$ where $\mathcal{V} = \{v_1, v_2, v_3, \cdots v_{N_g}\}$ is N_g geometrical data and $\Psi = \{\psi_1, \psi_2, \psi_3, \cdots \psi_{N_t}\}$ is N_t topological data. Since triangle is elementary polygon usually each topological data represents one triangle. Remember all polygons can be represented as a group of triangles. Multiple description for 3D graphics represents each geometrical data with two different descriptions. We assume all of topological data are available at the decoder.

4.3.2 Multiple description scalar quantization (MDSQ) for 3D graphics

For each geometrical data of 3D graphic we design two m-levels descriptions that complement each other. So if both channels work then the number of quantization

level at the encoder is

$$l = (2k+1)m - k(k+1) \tag{4.4}$$

where k means that the IA use a k off-diagonal matrix. In MDSQ for 3D graphics, at first, the minimum and maximum of each coordinate are determined for the uniform quantizer. Then the encoder decides about the step size Δ of uniform quantizer. Let assume the minimum and maximum of x, y, z directions are (x_{min}, x_{max}), (y_{min}, y_{max}), and (z_{min}, z_{max}), respectively. The three index numbers of nth vertex $v_n = \{x_n, y_n, z_n\}$ for MDC are given by

$$\begin{aligned} j_n^x &= \arg\min_i (x_n - x_{min} + \Delta_x \times i)^2 \\ j_n^y &= \arg\min_i (y_n - y_{min} + \Delta_y \times i)^2 \\ j_n^z &= \arg\min_i (z_n - z_{min} + \Delta_z \times i)^2 \end{aligned} \tag{4.5}$$

where Δ_x, Δ_y and Δ_z are step size of uniform quantizer having l quantization levels for x, y and z coordination, respectively.

In quantization procedure, depending on the quantization levels of a coordinate, two or more geometrical data can be merged to one geometrical data. In this case element polygon (triangle) changes to line or point. In index assignment procedure, using IA matrix (Figure 4.2(a)), from this indices of each geometrical data, the encoder generate two descriptions. Thus for each triangle six indices are generated. The MDC encoding is represented as $g(\psi) = \{(r_i^x, r_i^y, r_i^z), (c_i^x, c_i^y, c_i^z)\}$, $i = 1, 2, 3$. Two descriptions (r_i^x, r_i^y, r_i^z) and (c_i^x, c_i^y, c_i^z) are transmitted via different

physical channels. At the decoder, 3D graphic is regenerated using both descriptions if there is no problem in ether channels. Otherwise the decoder regenerates a coarse 3D graphic using the available description.

4.3.3 Multiple description vector quantization (MDVQ) for 3D graphics

In vector quantization for 3D graphics, for l level quantization, there are l^3 index numbers. The index number for nth vertex $v_n = \{x_n, y_n, z_n\}$ is

$$\lambda_n = \arg\min_i \|v_n - \hat{v}_i\|, \tag{4.6}$$

where $\hat{v}_i$ is the ith vertex in 3D rectangular structure having uniform step size. Figure 4.2(b) depicts the structure of uniform vector quantizer and IA mapping in 3D. After determining the index number for each vertex the IA which was introduced by [40] is followed to generate two descriptions. Thus, for each triangle the encoder generates six descriptions which are $g(\psi) = \{r_i, c_i\}$, $i = 1, 2, 3$.

In MDVQ two coarse lattices represents one fine lattice. The encoder finds a pair of index number in coarse lattices based on nearest neighbor search. When designing MDVQ, the coarse lattice should be geometrically similar to fine lattice. In MDVQ for 3D graphics, we find two nearest points from coarse lattices to represent one fine description in 3D vector quantizer structure. In Figure 4.2(b), the point **ae** in fine lattice is represented with two descriptions (description **A** and description **E**) in coarse lattices. Thus, the first letter of fine lattice consists of one

description and the second latter consists of another description. At the decoder, if both descriptions are available the exact position can be reconstructed. Otherwise the position is reconstructed with one description.

4.4 Side Decoder for MDC

Designing of multiple description decoder is based on the assumption that the encoder generates two descriptions. When decoder has both descriptions the difference between input source and reproduced one is minimum in term of mean squared error (MSE). With this condition we need to consider designing side decoder minimizing side distortion. In the following subsection we describe three different decoding method to minimize the side distortion when both descriptions are not available at the decoder. To design side decoder we assume only one description $r_i = a(q(x_i)) \in J_m$ ($c_i = a(q(x_i)) \in J_m$) is available at the decoder.

When we design side decoder there are three difference cases. First, there is no side information in either the encoder or decoder. Second, side information is available just at the decoder. Third, side information is available both at the encoder and decoder. In this chapter we consider two cases in which the side information is available at the decoder or not.

4.4.1 Side decoder for standard multiple description coding

If encoder use uniform scalar quantizer the output with j quantization level is simply $x_{min} + \frac{2j-1}{2}\Delta$, where Δ is the step size. If decoder has only one description r_i for input source x_i then the decoder can not use IA matrix to get the index of

quantization j_i. In IA matrix there are several indices of quantization for each r_i. One way to decode with multiple indices is considering all of indices with same weight. The decoder considers both description r_i and IA matrix together to find the index of quantization j_i. Then the approximation x_i using several j_i is

$$\hat{x}_i = x_{min} + \frac{1}{|\mathcal{C}_{r_i}|} \sum_{j \in \mathcal{C}_{r_i}} (j - 0.5)\Delta \tag{4.7}$$

where $\mathcal{C}_i$ is the ith column (row) vector of IA matrix and $|\mathcal{C}_i|$ is the size of $\mathcal{C}_i$. For example, when description 1 is available at the decoder, using IA matrix in Figure 4.1(b), $\mathcal{C}_2 = \{3, 6, 7, 9\}$ and $|\mathcal{C}_2| = 4$.

This side decoder consider all possible indices with same probability. This side decoding method is efficient if encoder generate two descriptions with high redundancy. But when IA uses big off-diagonal matrix the side decoder consider too many indices as candidate index.

4.4.2 Side decoder based on diagonal element

Another side decoding method without side information is the side decoder assuming that the encoder use diagonal matrix as IA matrix. Since the encoder does not use diagonal matrix as IA matrix, difference between the original and reconstructed data is unavoidable. The side decoder based on diagonal element has much simpler structure than the side decoder of standard MDC. In this decoding method, the decoder assumes the lost description is same with the available description. For example, if only r_i is available at the decoder the decoder generates

the index of quantization as $j_i = a^{-1}(r_i, r_i)$. On the contrary, if only c_i is available at the decoder the index of quantization is $j_i = a^{-1}(c_i, c_i)$. Using the index of quantization j_i the decoder approximates x_i as $\hat{x}_i = x_{min} + \frac{2j_i-1}{2}\Delta$.

Although this decoding method is simple there is high probability of generating wrong index number. As the redundancy of IA matrix decrease this probability increases. For example when we use eight-by-eight IA matrix with two off-diagonal the possibility of correct quantization index number is $\frac{1}{5}$. But if we use eight-by-eight IA matrix with one off-diagonal the possibility increases to $\frac{1}{3}$.

4.4.3 Side decoder based on distribution of input source

In the standard side decoder for MDC, the decoder assumes that all indices of quantization have the same probability. When input source has non-uniform distribution such as Gaussian and Laplacian this assumption may not be true. Most real data have non-uniform distribution. When the input source has non-uniform distribution, if the decoder knows the probability of each index of quantization, the decoder can reduce side distortion using this information. The probability of the index of quantization can be transmitted to decoder as side information. Also we assume there is no channel error during transmission of the side information. Lets assume for each index of quantization $j \in J_m$ the probability $p(j)$ is $P(j = q(X))$.

When only one description is available (r_i) the decoder get a vector $\mathcal{C}_{r_i} = \{c_1, c_2, \cdots c_n\}$. Using the vector of index number $\mathcal{C}_{r_i}$ the decoder reproduce input source data as

$$\hat{x}_i = \sum_{j \in \mathcal{C}_{r_i}} (x_{min} + (j - 0.5)\Delta) \times \frac{p(j)}{\sum_{l \in \mathcal{C}_{r_i}} p(l)}. \tag{4.8}$$

Table 4.1: The number of vertices and faces of test 3D graphics.

	Sculpture	Bunny	Shoe	Venus
No. of vertices	25,386	35,947	78,239	711
No. of faces	50,780	69,451	156,474	1,396

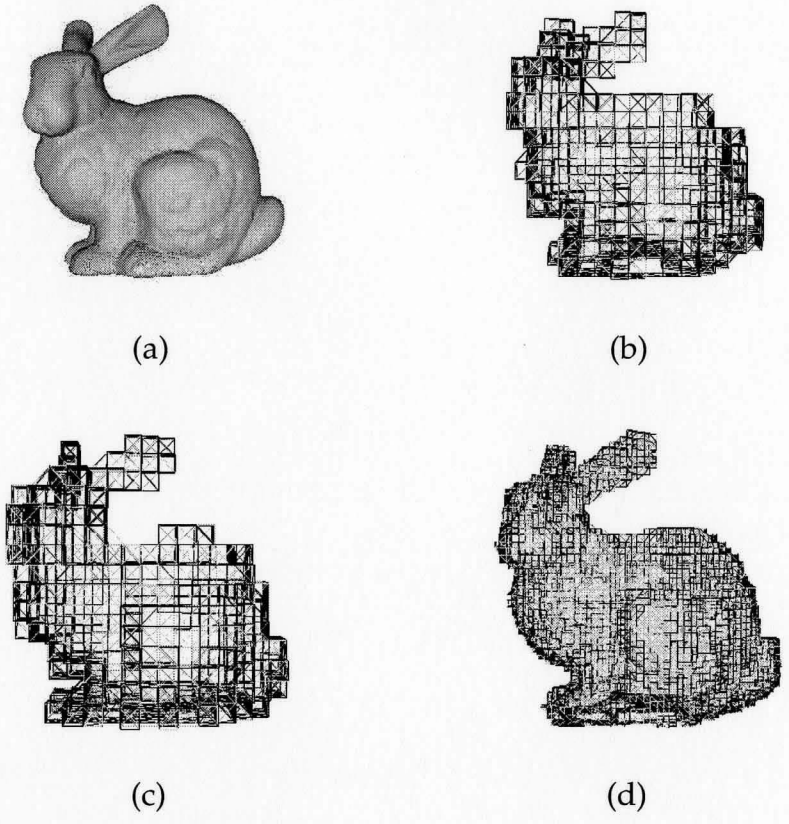
This method considers all possible indices as well as the probability of each index. Although there is still difference between original and reproduced data, this method tries to minimize the difference when input source has non-uniform distribution.

4.5 Experimental results

4.5.1 MDSQ and MDVQ for 3D graphics

At first we investigate the performance of MDC for 3D graphics using test modes: Sculpture, Bunny, Shoe and Venus. We use both scalar and vector quantizer. The decoder decodes received data without side information. The number of vertices and faces of our test models are given in Table 4.1. For MDSQ we used four different IA matrices with different quantization levels.

Figure 4.3 shows the performance of MDSQ for 3D graphic. For IA matrix at the encoder we use $m = 64$ and $k = 1$. Thus every input data is quantized using uniform qunatizer having 190 quantization levels. Figure 4.3(a) shows original Bunny model used for MDSQ. Figures 4.3(b) and 4.3(c) are the reproduced Bunny with description 1 and 2 at the decoder, respectively. Although the shape of the model is not as clear as reproduced model with two descriptions (see Figure 4.3(d)) we



(a) (b)

(c) (d)

Figure 4.3: MDSQ performance with $m = 64, k = 1$: (a) Original Bunny, (b) reproduced model with description 1, (c) reproduced model with description 2 and (d) reproduced model with both descriptions.

Table 4.2: The performance of MDVQ with different quantization levels.

		Sculpture	Bunny	Shoe
$l = 25$	md	7,108	4,411	5,780
	md1	276	192	194
	md2	296	168	178
$l = 50$	md	21,310	16,677	22,716
	md1	1,228	758	894
	md2	1,260	752	918
$l = 75$	md	33,192	33,829	48,192
	md1	2,802	1,738	2,004
	md2	2,840	1,721	2,168
$l = 100$	md	40,856	51,255	79,098
	md1	4,710	3,019	3,954
	md2	4,814	3,006	3,926

can easily find that the reproduced model looks like Bunny. Figure 4.3(d) shows how MDSQ using simple uniform scalar quantizer reproduces better 3D graphic models when the decoder received both descriptions than 3D graphic models reproduced with one description.

At the encoder of MDVQ we adapt 25, 50, 75 and 100 quantization levels for each coordinate. Table 4.2 summarize the number of faces of regenerated Sculpture, Bunny and Shoe model when encoder use different quantization levels. In case of Sculpture model, when we change quantization levels form 100 to 25, the original model can be represented with 80.46, 65.36, 42 and 14% of the original number of faces. If we use same quantization level for Shoe model the original model is represented with 50.55, 30.80, 14.52 and 3.69 % of the original number of faces. MDVQ for 3D graphics is more useful when the model has more faces.

Figure 4.4 show the performance of MDVQ for 3D graphics. Figure 4.4(a) represents original Bunny model. Figures 4.4(b) and 4.4(c) are reproduced Bunny model

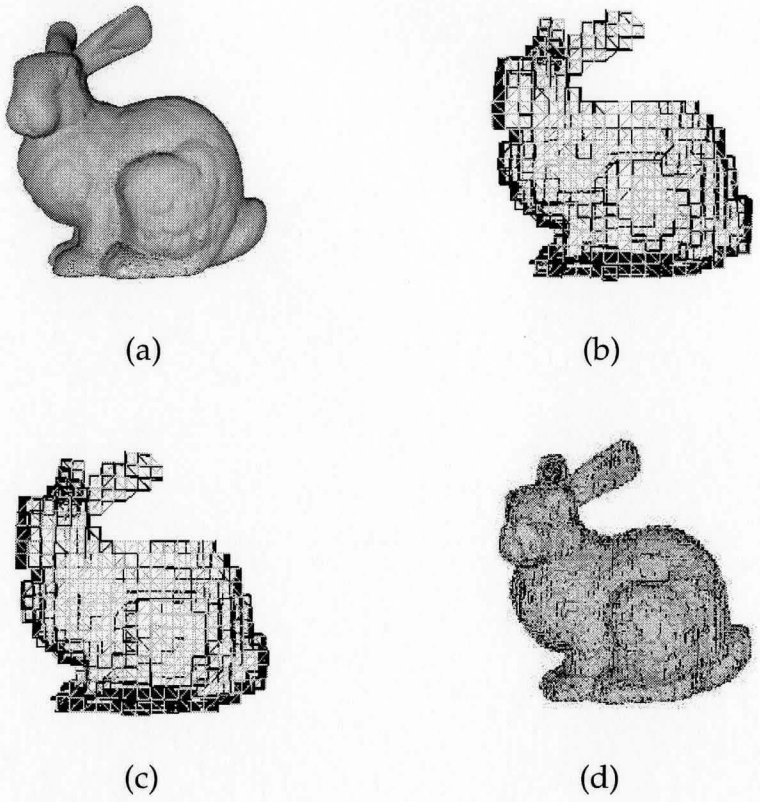


Figure 4.4: MDVQ performance with 3D graphics model of Bunny $l = 100$: (a) Original Bunny, (b) reproduced model with description 1, (c) reproduced model with description 2 and (d) reproduced model with both descriptions.

at the side decoders when just one description is available at the decoder. Figure 4.4(d) depicts the reproduced Bunny model when both descriptions are available at the decoder. The quantization level at the encoder is 100.

4.5.2 The performances of different side decoders

To verify the the performances of three different side decoders we simulate the algorithm using different sources. Especially we investigate the performance of side decoder when side information is available. Scalar quantizer is adopted to quantize the input source. The number of quantization levels depends on the bit-rate. Using n-bits for each description the size of IA matrix is 2^n-by-2^n. Then the number of quantization levels is $l = (2k+1)2^n - k(k+1)$ where k means that the IA matrix is k-off diagonal matrix. For input source we use random data having Uniform, Gaussian and Laplacian distribution. Also 3D graphic data is used to investigate the performance of the side decoder. The step size of quantizer is $\Delta = (x_{max} - x_{min})/l$. The maximum bound x_{max} and minimum bound x_{min} are

$$\begin{aligned} x_{max} &= \mu + \gamma \times \sigma \\ x_{min} &= \mu - \gamma \times \sigma \end{aligned} \tag{4.9}$$

where μ and σ is mean and standard deviation of input source, respectively, and γ is a constant deciding the boundary between overload and granular probability area.

We use mean squared error as distortion criteria. The central and side distortion are $d(X, \hat{X}_0) = mse(X, \hat{X}_0)$ and $d(X, \hat{X}_{1,2}) = \frac{mse(X,\hat{X}_1)+mse(X,\hat{X}_2)}{2}$, respectively.

Figure 4.5 shows the performances of three side decoders in terms of MSE vs bit-rate when IA has different off diagonal matrix. In Figure 4.5(a) we compared the distortion of three side decoders and central decoder when IA is one off-diagonal matrix. When $k = 1$, since there is high redundancy between two descriptions the performance of side decoder with diagonal element is similar to the performances of the other side decoders. Among three side decoder the side decoder with probability information has the best performance. Figure 4.5(b) depicts the performance when encoder uses an IA matrix with two off-diagonal. The performance of side decoder with diagonal element is not as good as the performance when encoder uses an IA matrix with one off-diagonal. The performance of side decoders is not as good as the performance of central decoder. The reason is that the redundancy of two description is decreased.

Figure 4.6 shows the performance of standard and proposed side decoders for source with different distribution. In Figures 4.6 and 4.7 the MSEs are calculated in term of several different bit per sample (bps). Figures 4.6(a) and 4.6(b) are the simulation results when input data has Uniform and Laplacian distribution, respectively. Two-off diagonal IA matrix is used to represent quantized index number. To decide maximum and minimum quantization boundary we use $\gamma = 2.5$ and 2.8 for Uniform and Laplacian distribution source, respectively. When input source has Uniform distribution the performance of the side decoder based on probability information is almost same with the performance of standard MDC side decoder. If input source has non-uniform distribution the performance of proposed side decoder is better than the performance of standard MDC side decoder as can be see in Figure 4.6.

Figure 4.7 is the result when we use 3D graphic data as input source. Uniform scalar quantizer is used in encoder to reduce the bit rate. We change bit-rate from two bits per sample to twelve bits per sample. In case of 3D graphic data γ is 2.0. From Figure 4.7 we can easily find that the proposed side decoder based on probability information has better performance than standard MDC side decoder. When the bit-rate is increased the performance of side decoder based on diagonal element is similar to the performances of the other two decoders. Figure 4.7(a) shows the simulation result when Bunny model is used as input source of MDC. Diagonal element based side decoder has the worst performance. Probability information based side decoder has better performance than standard MDC decoder. When decoder receive two descriptions the decoder reproduces Bunny with the best quality (the central decoder case).

Figures 4.7(b) and (c) are simulation results when source inputs are Venus and Shoe, respectively. At the low-bit area (when the bit-rate is less than six) the proposed probability information based side decoder has better performance than the performance of the decoder using both descriptions. Figure 4.7(d) depicts the performance of side decoder when Sculpture model is used as input source. In case of Sculpture model the proposed probability information based side decoder has better performance than the performance of decoder with both descriptions in all bit-rate range. The performance is similar when bit-rate is increased. Thus, in case of Sculpture the proposed side decoder has better performance in the high bit-rate area. In case of Shoe and Venus the proposed side decoder has better performance in low bit-rate than high bit-rate area.

Figure 4.8 shows the reproduced 3D graphic at the decoder using different

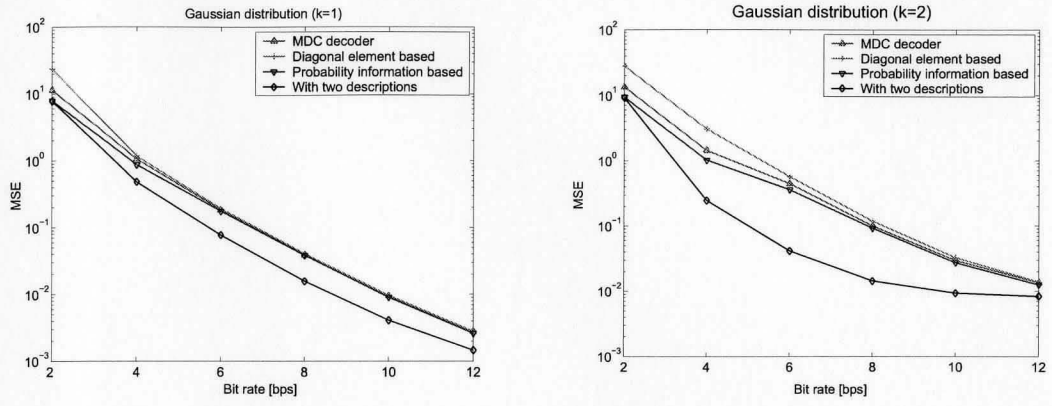


Figure 4.5: The performances of side decoder when encoder use different off-diagonal matrix: (a) $k = 1$, (b) $k = 2$.

number of descriptions and different side decoders. The Bunny having 35,947 vertices and 69,451 faces is used as input source. The bit-rate for each descriptor is four bits per sample and k is two. So quantization levels are 16 and 58 for side and central decoder, respectively. Figure 4.7(a) depicts the 3D graphic when decoder use both descriptions. The resolution of the 3D graphic is very fine. We almost recognize that the reproduced 3D graphic looks like original Bunny. Figure 4.7(b)-(d) are final results when decoder use just one description and side decoder. Although the regenerated 3D graphic is not as good as 3D graphic with two descriptions we still can assume the 3D graphic as Bunny.

Even though there are different MSE on the results of side decoders the regenerated 3D graphics are very similar to each other. Thus, we can use side decoder based on diagonal element to reduce computational complexity and the bit-rate. If channel capacity is enough the side decoder based on probability information is more desirable than other decoders.

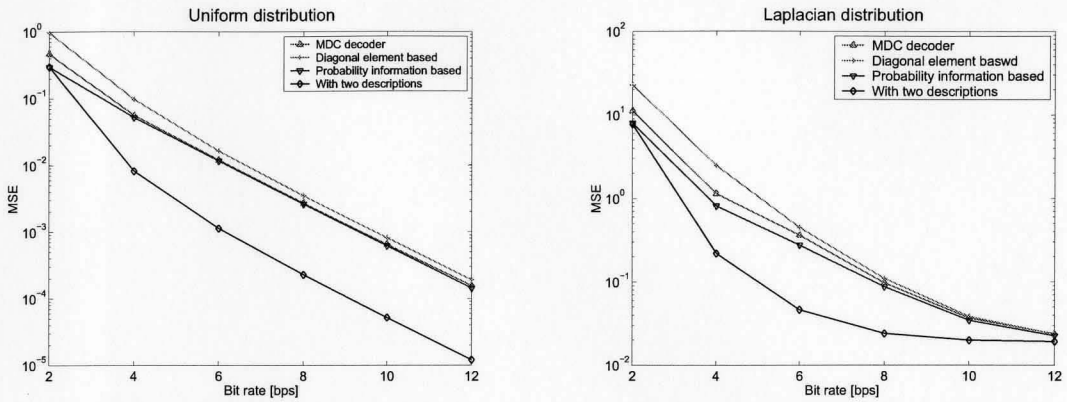


Figure 4.6: The performance of side decoder with different source input: (a) Uniform distributed data and (b) Laplacian distributed data.

4.6 Error concealment for moving 3D graphic object

Communication of moving 3D graphic model over error prone channel is an another issue. Since there are high temporal redundant between two successive 3D graphic model, the motion information can be investigated to conceal the erroneous 3D graphic data. Since human face is a good example of 3D graphic we concentrate on human face for error concealment of moving 3D graphic model. The proposed idea can be extended to other moving 3D graphics.

Communication of human faces is essential in many multimedia applications such as videoconferencing, distance learning and virtual reality. All of these applications demand rendering of a realistic human face for efficient and comfortable face-to-face communication. Human face communication can be divided into two groups: natural and synthetic. Compression of natural facial video can be achieved

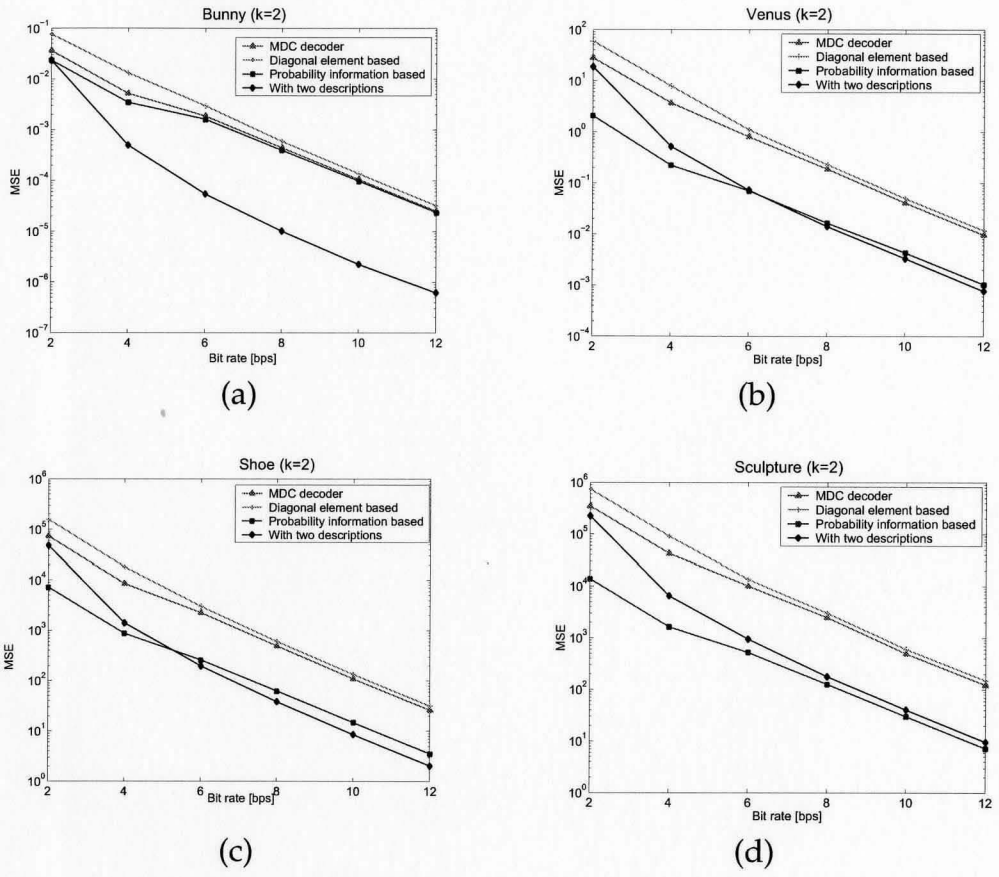


Figure 4.7: The performance of side decoder with 3D graphic data: (a) Sculpture, (b) Bunny, (c) Shoe and (d) Venus.

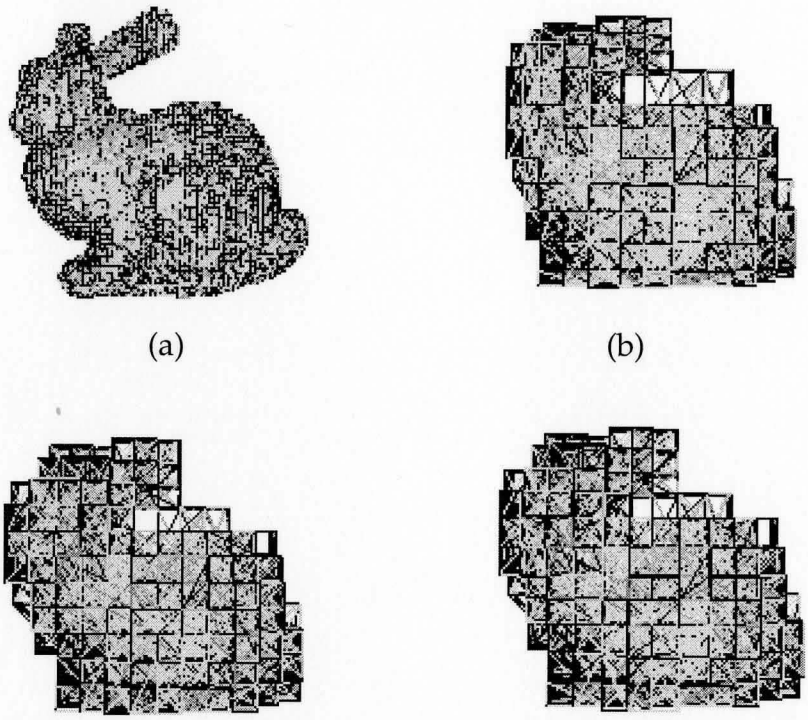


Figure 4.8: Decoded 3D graphic with different descriptions and side decoders: (a) with two descriptions using central decoder, (b) one description using MDC side decoder, (c) one description using side decoder based on diagonal element and (d) one description using side decoder based on probability information.

by using one of many video compression approaches for example, the hybrid discrete cosine transform (DCT) and motion compensation coders which are adapted by video coding standards such as H.263, MPEG-2, and MPEG-4 [41,42].

Although these compression algorithms reduce the amount of data used to represent natural face, data rate may still be too high for some applications. In this case, a synthetic face generated from a natural face is an alternative. Representing faces using synthetic methods requires less data compared to natural representation and therefore under limited bandwidth circumstances, synthetic face representation may be preferable. Although natural representation of an object with 2D data provides acceptable results representing an object with 3D data is more realistic. Waters [43] developed a muscle-based facial model for realistic facial animation.

Several compression methods for synthetic facial representations have been developed for very low bit-rate applications [44]. For example, masking and spatial correlation methods are employed to reduce the number of facial parameters and hence achieve very low bit-rate facial animation [45,46]. The masking method compresses only that subset of facial parameters in the current frame which is different from that of the previous frame instead of compressing all of the facial parameters. Ahlberg and Li [46] used a facial action basis function to reduce spatially correlated facial animation data. In these compression algorithms, every compressed data represents several facial animation data.

Thus, each compressed facial data is important for rendering facial animation at the receiver. Distortion in rendered facial animation will occur at the receiver terminal if compressed data is lost or changed during transmission. Under this

circumstance, error concealment is recommended to reduce the visual artifacts caused by transmission errors. Varakliotis *et al.* [47] introduced an error concealment method for facial animation parameters (FAPs) in MPEG-4 environment. Varakliotis' algorithm uses a tracking curve at the receiver to interpolate the position and predict the velocity of the animated vertices for missing frames. Ishikawa *et al.* [48] proposed a method for estimation of facial muscle parameter from 2D natural images.

In this section we propose an error concealment method for synthetic facial data based on the linear interpolation of muscle data. For linear interpolation, a facial expression flow which has start expression and end expression is determined using dominant muscle data. Our error concealment method is novel in that it employs both forward error concealment and post processing. In forward error concealment, in addition to muscle data, the transmitter sends side information of the facial expression flow. To determine the facial expression flow, the transmitter classifies the current facial expression based on the dominant muscle data which are those muscle data with maximum change between two successive frames. In error concealment by post processing, the decoder uses facial data in the frame before an erroneous frame to estimate the missing facial data.

4.6.1 Background

In 3D representation an object consist of geometrical and topological data set. The geometrical data describe the information of three-dimensional points (we represent this information as x, y and z of each point). The ith geometrical data is expressed with the vector $v_i = \{x_i, y_i, z_i\}$. Usually each geometrical data is called a

vertex or node of an object. The topological data describes a polygon based on its vertices. Each topological data represents one polygon. Thus, the topological data is a sequence of the index number of geometrical data. The size of each topological data is the same as the number of the vertices of polygon represented by topological data. Since triangle is the elementary unit of a polygon, jth topological data consist of three geometrical data (vertices) and is expressed as $\psi_j = \{v_l, v_m, v_n\}$ where v_l, v_m and v_n are the vertices of triangle ψ_j. Then one object $\mathcal{S}$ in 3D is expressed with a set of triangles as

$$\mathcal{S} = \bigcup_{j=1}^{N} \psi_j, \tag{4.10}$$

where N is the number of triangle consisting an object $\mathcal{S}$. In the following subsections we use this three-dimensional structure to explain facial expression.

4.6.2 Facial Animation and Facial Expression

The 3D face model consists of a number of non-overlapped triangles. As a matter of convenience in representing moving facial expression the facial data set (F) and facial object ($\mathcal{S}(F)$) in three dimension are expressed as

$$\begin{aligned} F &= \{v_i, \psi_j | i = 1, 2, \cdots, N_G, j = 1, 2, \cdots N_T\} \\ \mathcal{S}(F) &= \bigcup_{j=1}^{N_T} \psi_j \end{aligned} \tag{4.11}$$

where N_G and N_T represent the number of geometrical and topological data, respectively.

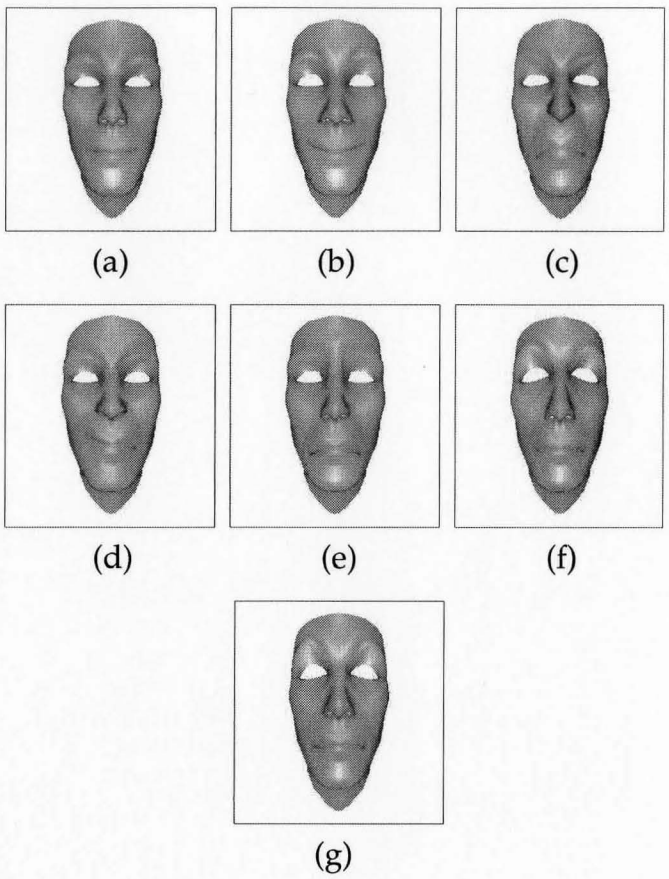


Figure 4.9: Seven facial expressions: (a) natural, (b) happiness, (c) anger, (d) disgust, (e) surprise, (f) sadness, (g) fear.

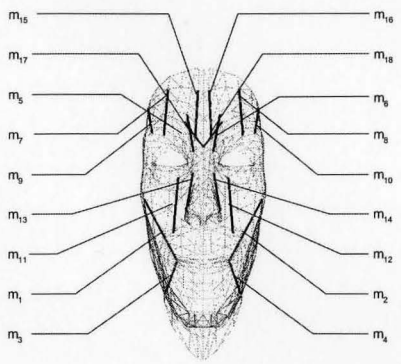


Figure 4.10: Facial mesh and muscle representation in 3-D.

In humans, there are many facial expressions and defining all of them is difficult. To represent emotional expression efficiently, six basic facial expressions (plus natural expression) are widely used. They are happiness, sadness, surprise, fear, anger and disgust. Figure 4.9 shows rendered six basic expression and natural face for a three-dimensional face model. During last few decades, many researchers have studied the recognition of human facial expression. Some of these studies are based on natural images and others are synthetic based [49]. G. Donato *et al.* [50] performed a survey and comparison of techniques for facial expression recognition of image-based facial expression. Face muscle data are a useful tool for facial animation. Facial expression classification techniques using muscle data are described by Ishikawa *et al.* [48] and Ohta *et al.* [51].

Although expressing the face using synthetic method reduces the data, it is necessary to reduce the data more for limited bandwidth applications. In facial animation using muscle data, a combination of muscle data animates a specific facial expression [48]. The expression of the face changes by changing the position of

vertices of triangles representing the face. The position of vertices of each triangle is determined by a combination of muscles [52]. Therefore

$$v_i = \sum_{k=1}^{N_M} \alpha_{i,k} m_k \tag{4.12}$$

where v_i is the position of ith vertex, N_M is the number of muscle data, m_k is kth muscle data and $\alpha_{i,k}$ is the coefficient of the kth muscle data related to the ith vertex.

Figure 4.10 shows a 3D face model consisting of 876 triangles and 18 muscles. By representing facial expression using several muscle data instead of using all of the geometrical data, facial animation is rendered with a reduced data set. So it is possible to control facial expression with muscle data after sending all of the geometrical and topological data in initial stage. For example, the facial expression of Figure 4.10 can be represented with 18 muscle data instead of using the geometrical and topological data of 876 triangles. Usually, fewer muscle parameters, typically 18, are enough to animate a facial expression. Table 4.3 lists the muscle names which are usually used [53].

4.6.3 Error prone channels, transmission errors and error concealment

In data transmission over error prone channel, errors are inevitable. The error rate of these channel is time varying. The underlying processes that lead to data losses over error prone channel is complex [54]. However Gilbert-Elliott model gives us good approximations for loss behavior of error prone channels. Thus, a simple

Table 4.3: Muscle number and name

Muscle number	Muscle name
m_1	Left zygomatic major
m_2	Right zygomatic major
m_3	Left angular depressor
m_4	Right angular depressor
m_5	Left frontalis inner
m_6	Right frontalis inner
m_7	Left frontalis major
m_8	Right frontalis major
m_9	Left frontalis outer
m_{10}	Right frontalis outer
m_{11}	Left labi nasi
m_{12}	Right labi nasi
m_{13}	Left inner labi nasi
m_{14}	Right inner labi nasi
m_{15}	Left lateral corigator
m_{16}	Right lateral corigator
m_{17}	Left secondary frontalis
m_{18}	Right secondary frontalis

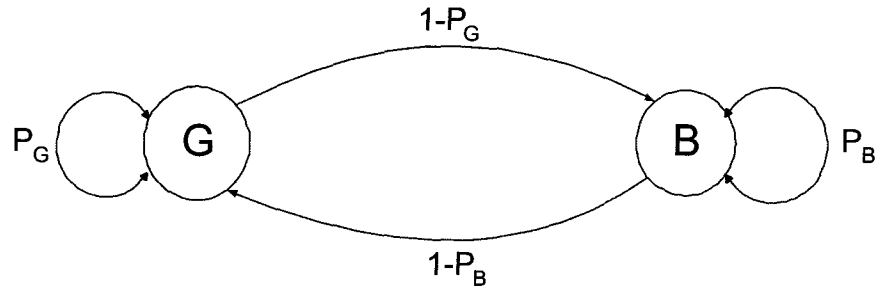


Figure 4.11: Gilbert-Elliott channel model.

time varying channel can be simulated using Gilbert-Elliott model. Figure 4.11 shows Gilbert-Elliott channel model. In Figure 4.11 G and B denote the good and bad state of channel, respectively. Also the channel error probabilities in good (G) and bad (B) states are represented with P_G and P_B, respectively.

The Gilbert-Elliott channel model has an error-free interval distribution. The error-free interval is defined with the length of bit l_s (i.e, the number of bits between two consecutive errors) and the length of packet size l_p. Then the probability of error in the current bit is

$$P_e = \begin{cases} (1 - P_B)(P_G)^{l_p - l_s} & \text{error on the previous bit} \\ (P_G)^{l_p - l_s} & \text{otherwise} \end{cases} . \tag{4.13}$$

More detail on Gilbert-Elliott channel can be found in [55, 56].

In representing facial animation using muscle data, usually one muscle datum relates to many points of a rendered face. Missing one muscle datum therefore may cause critical deformation of rendered facial animation at the receiver. Thus, in an error prone channel application, error concealment is necessary to achieve good

visual quality. Error concealment techniques are classified into three categories: (1) forward error concealment, (2) error concealment by post processing, and (3) interactive error concealment.

In the forward error concealment, the source coding algorithm is designed either to minimize the effects of transmission errors or to make the error concealment task at the decoder more effective. These techniques usually increase the overhead in terms of the overall bitstream size; hence, part of the coding efficiency achieved by compression is lost. In the error concealment by post processing, the decoder fulfills the task of error concealment. The objective of error concealment by post processing is to remove visually annoying artifacts by estimating the lost information without relying on additional information from the encoder.

In the interactive error concealment, it is assumed that a backward channel from the decoder to the encoder is available and the encoder and decoder work cooperatively to minimize the impact of the errors. An example of such a technique is Automatic Retransmission Request (ARQ). Retransmission has generally been considered unacceptable for real time video applications because of the associated delay. Another disadvantage of retransmission is that it may increase the loss rate by adding more traffic on the network [57].

4.7 Proposed error concealment method

In this section, we describe the proposed error concealment algorithm in detail. At first, we explain the procedure of the tracking of facial expression flow. Then the side information for forward error concealment is mentioned. The error detection and concealment methods are explained in the following Section. Figure 4.12

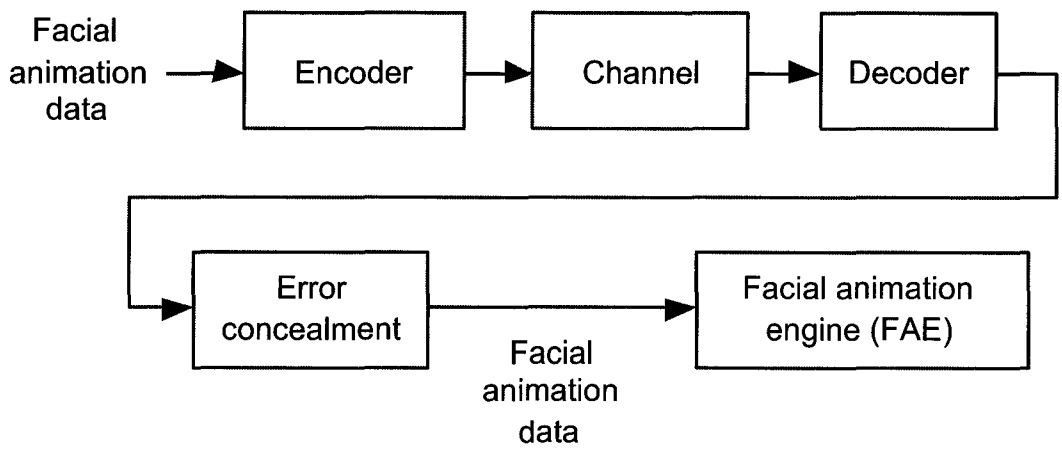


Figure 4.12: General error concealment processing.

shows the overall block diagram of the synthetic face communication system. At the setup stage, 3D mesh data (a geometrical and topological data set which are F in Equation 4.11) are sent to the receiver to generate a 3D face model using a facial animation engine (FAE).

After the completion of the setup stage, muscle data is sufficient to animate the face and change the expression. If compressed muscle data is affected during transmission, the error concealment block is activated to reduce the visual effects of error. Our error concealment method employs both forward error concealment and error concealment by post processing. If the channel has enough bandwidth, forward error concealment is adapted; otherwise error concealment is done by post processing.

In forward error concealment, in addition to muscle data, the transmitter sends side information describing facial expression flow. Figure 4.13(a) shows the proposed encoding process for forward error concealment. In Figure 4.13(a) M_n is the

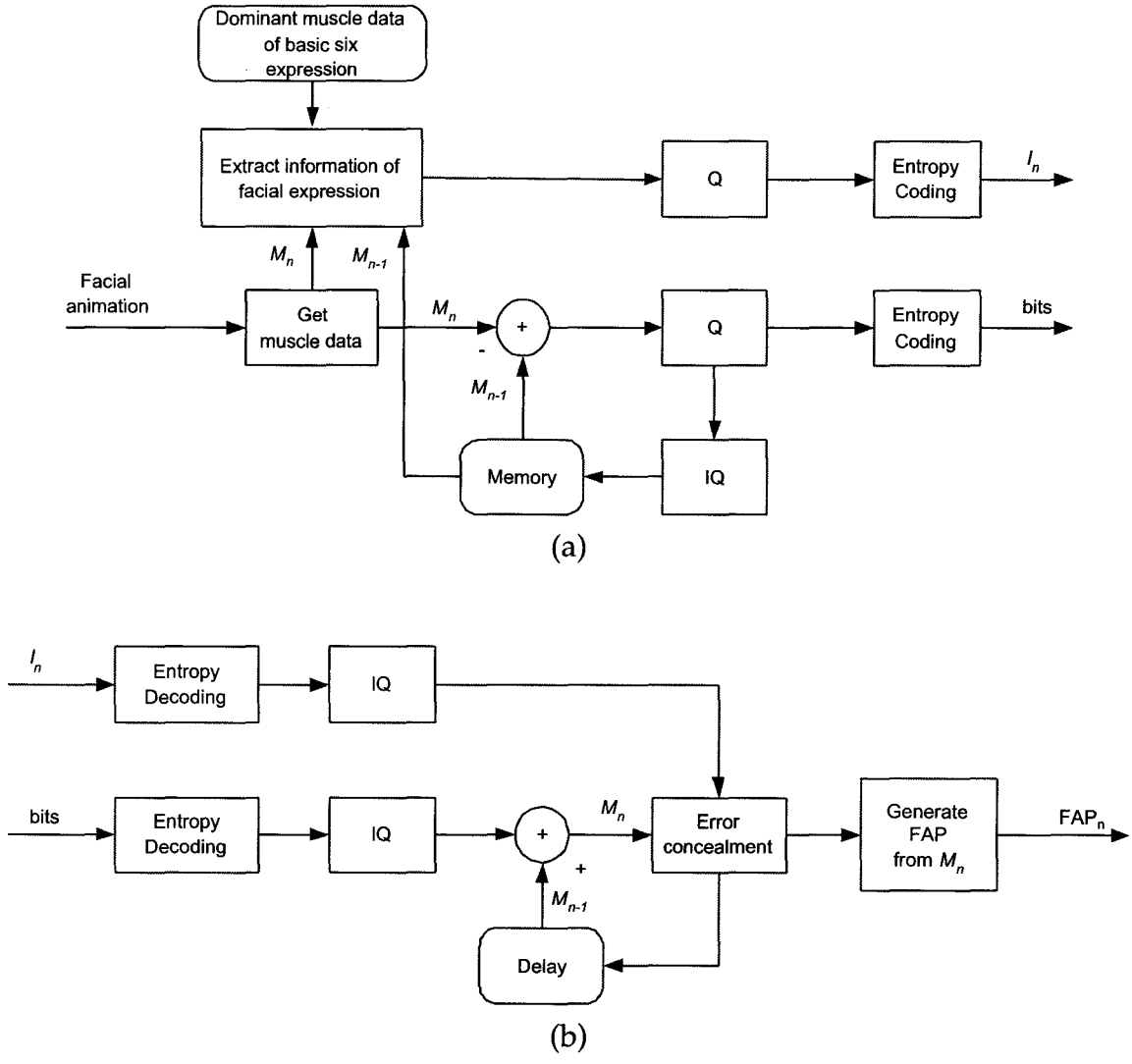


Figure 4.13: Block diagram of proposed error concealment method: (a) encoding process (transmitter) and (b) decoding process (receiver).

muscle data set of the nth frame (e.g., $M_n = \{m_k(n), k = 1, 2, \cdots, N_M\}$) and I_n is side information about facial expression flow. As can be seen in Figure 4.13(a) the muscle data is compressed using differential coding, quantization and entropy coding. The side information (I_n) consists of two parts: (1) start expression (SE) and end expressions (EE) which can be any of seven expressions (e.g., happiness, sadness, surprise, fear, anger and disgust and natural) and (2) muscle data ratio (β_n). To extract the side information the encoder tracks the facial expression flow using dominant muscle data as will be explained in Section 4.7.1.

4.7.1 Tracking of facial expression

For generation of side information employed in forward error concealment, we propose a new and simple method for tracking and classification of synthetic facial expression. Since we use seven expressions, there is $2 \times_7 C_2 = 42$ possible different expression flows. To initially classify the facial expression, the transmitter defines the dominant increased and decreased muscle data of 42 possible expression changes. That is:

$$\mathcal{D}_{N_L}(n) = \mathfrak{S}_L(d_1(n), d_2(n), ..., d_{N_M}(n)) \tag{4.14}$$

where

$$d_k(n) = m_k(n) - m_k(n-1).$$

$\mathcal{D}_{N_L}(n)$ is the set of dominant muscle data for nth frame, $m_k(n)$ is the kth muscle data of nth frame, N_M is the number of muscle data and $\mathfrak{S}_L(\cdot)$ is the operator that

sorts the data in decreasing order and returns the indices to N_L largest (dominant increased) and N_L smallest (dominant decreased) values. The transmitter then compares $\mathcal{D}_{N_L}(n)$ with data in an expression information table (EIT) and finds a match.

The EIT consists of dominant muscle data for different facial expressions changes. To determine sets of dominant muscle data for different facial expressions changes and thus build the EIT used in matching, the difference of muscle data between two expressions is

$$d_k^{(a,b)} = m_k^a - m_k^b \tag{4.15}$$
$$\{a, b\} \in \{N, H, A, D, Sad, Sur, F\}$$

where m_k^a is kth muscle data of facial expression a, a (and b) is one of seven expression; N: natural, H:happiness, A: anger, D: disgust, Sad: sadness, Sur: surprise and F: fear. The muscle data set of seven expressions is predefined by rendering facial animation. Using the difference muscle data, dominant muscle data of two facial expressions are formed as

$$\mathcal{D}_{N_L}^{(a,b)} = \mathfrak{S}_L(d_1^{(a,b)}, d_2^{(a,b)}, ..., d_{N_M}^{(a,b)}) \tag{4.16}$$

where $\mathcal{D}_{N_L}^{(a,b)}$ is the dominant muscle data for a facial expression starting with expression a and ending with expression b. So in Equation (4.16), when expression flow is form a to b, $d_1^{(a,b)}$ represents the most dominant increased muscle data. Table 4.4 shows an example of the EIT when $N_L = 5$ and $N_M = 18$.

Using this table, the facial expression flow of a given muscle data sequence is

Table 4.4: Dominant muscle data for change of the facial expression

Expression	Dominant muscle data($D_n^{(a,b)}, n = 5$)	
(a, b)	Increase	Decrease
N,H	$m_1 m_2 m_6 m_5 m_{10}$	$m_{18} m_{17} m_{16} m_{15} m_{14}$
N,A	$m_9 m_{10} m_{12} m_{11} m_{15}$	$m_8 m_6 m_2 m_1 m_7$
N,D	$m_{16} m_{15} m_{12} m_{10} m_{11}$	$m_{18} m_{17} m_8 m_7 m_6$
N,Sad	$m_6 m_5 m_3 m_4 m_{12}$	$m_{18} m_{17} m_{16} m_{15} m_{10}$
N,Sur	$m_5 m_6 m_{10} m_{18} m_7$	$m_{16} m_{15} m_{13} m_{12} m_{11}$
N,F	$m_1 m_7 m_2 m_3 m_4$	$m_{14} m_{13} m_{16} m_{15} m_{12}$
H,A	$m_9 m_{12} m_{11} m_{10} m_{15}$	$m_2 m_1 m_6 m_5 m_8$
H,D	$m_{16} m_{15} m_{12} m_{10} m_{11}$	$m_6 m_2 m_5 m_8 m_7$
H,Sad	$m_6 m_5 m_3 m_4 m_{12}$	$m_2 m_1 m_{10} m_8 m_7$
H,Sur	$m_{18} m_5 m_{17} m_3 m_4$	$m_2 m_1 m_{16} m_{15} m_{13}$
H,F	$m_3 m_4 m_{17} m_7 m_8$	$m_6 m_5 m_2 m_1 m_{10}$
A,D	$m_{16} m_{15} m_1 m_{12} m_2$	$m_9 m_{14} m_{11} m_3 m_{13}$
A,Sad	$m_6 m_5 m_4 m_1 m_2$	$m_{10} m_9 m_{11} m_{15} m_{16}$
A,Sur	$m_5 m_6 m_7 m_8 m_{17}$	$m_{12} m_{11} m_9 m_{15} m_{16}$
A,F	$m_1 m_2 m_7 m_8 m_6$	$m_{10} m_9 m_{12} m_{11} m_{15}$
D,Sad	$m_6 m_5 m_3 m_4 m_7$	$m_{16} m_{15} m_{10} m_{12} m_{11}$
D,Sur	$m_5 m_6 m_{18} m_7 m_{17}$	$m_{16} m_{15} m_{12} m_{11} m_{10}$
D,F	$m_7 m_3 m_8 m_2 m_{17}$	$m_{16} m_{15} m_{12} m_{10} m_{11}$
Sad,Sur	$m_{10} m_{18} m_7 m_{17} m_8$	$m_6 m_5 m_{12} m_4 m_3$
Sad,F	$m_1 m_7 m_2 m_8 m_{17}$	$m_6 m_5 m_{12} m_4 m_3$
Sur,F	$m_1 m_2 m_{11} m_{12} m_{15}$	$m_5 m_6 m_{10} m_{18} m_{17}$

(N: Natural, H: Happiness, A: Anger, D: Disgust, Sad: Sadness, Sur: Surprise, F: Fear)

determined as follows. From the given muscle data sequence, the transmitter first determines dominant muscle data using Equation (4.14). After determining the dominant muscle data, a matching process is done using information in Table 4.4 to choose the start expression and the end expression. For example, assuming a specific sequence, the transmitter finds muscle data m_9, m_{12}, m_{11}, m_{10} and m_{15} as dominant increased and m_2, m_1, m_6, m_5 and m_8 as dominant decreased muscle data. By referring to Table 4.4 the transmitter determines that the expression is changing from happiness to anger. In contrast, if the dominant increased data are m_2, m_1, m_6, m_5 and m_8, and decreased muscle data are m_9, m_{12}, m_{11}, m_{10} and m_{15}, the expression is changing from anger to happiness. In this way the transmitter determines the start and end expressions (SE and EE). Increasing the value of N_L increases the accuracy of our classification algorithm. The index of the match in EIT consists the first part of side information.

The second part of side information is the muscle data ratio. After determining the start and end expressions the transmitter calculates muscle data ratio

$$\beta_n = \frac{m_d(n) - m_d^{SE}}{m_d^{EE} - m_d^{SE}} \tag{4.17}$$

where m_d is the most dominant muscle data. Then, SE, EE and β_n form the side information I_n employed in forward error concealment.

4.7.2 Error detection and concealment

If error occurs during data transmission, distortion of facial animation may start and propagate until the receiver gets a new full facial data set. To minimize this

distortion, the receiver must constantly check the received muscle data set for the presence of error. Here we propose a mechanism for error detection. Two kinds of transmission error are considered: (1) data loss and (2) bit alteration. In the case of data loss, it is easy to detect the occurrence of error because the receiver knows how many muscle data are needed to render facial animation. To detect bit errors caused by bit alteration automatically, we define the maximum and minimum of the acceptable range of change of muscle data

$$\begin{aligned} T_{max} &= \max\{\mathcal{A}_x(d_k(n)), \mathcal{A}_n(d_k(n))\} \times \gamma \\ T_{min} &= -T_{max} \end{aligned} \tag{4.18}$$

where the function $\mathcal{A}_x(.)$ and $\mathcal{A}_n(.)$ give absolute maximum and minimum value, respectively, and $d_k(n)$ is the kth difference muscle data as defined by Equation (4.15) and γ is a constant that determines the margin for the maximum and minimum threshold.

Increasing γ makes the algorithm more generous to error. Using the above equations the range of allowed muscle data of the current frame depends on muscle data of the previous frame. This is reasonable since facial expressions usually do not change abruptly. Whenever the difference between current muscle data and previous muscle data is larger than T_{max} or smaller than T_{min}, or there is no muscle data, the receiver treats the muscle data as erroneous. Algorithm 3 describes the operation of error detection and concealment which works at the error concealment block of the decoder (Figure 4.13(b)).

Figure 4.13(b) shows the decoding process that includes error concealment. Whenever the receiver detects an error in the data of a frame, it checks whether

Algorithm 3 Error detection and concealment

1 Read differential muscle data set of nth frame

$$M_n = \{d_k(n) | k = 1, 2, \cdots, 18\}.$$

2 For each kth differential muscle data, check whether there is an error by checking

$$d_k(n) > T_{max} \text{ or } d_k(n) < T_{min}.$$

(1) If error exists and there is side information I_n:

- Do error concealment using preprocessing based error concealment:

$$m_k(n) \leftarrow \beta_n(m_k^{EE} - m_k^{SE}) + m_k^{SE}$$

(2) If error exists but there is no side information I_n:

- Do error concealment using post processing based error concealment.
- Update kth differential muscle data using the data of previous frame:

$$d_k(n) \leftarrow d_k(n-1).$$

- Update kth muscle data using kth differential muscle data:

$$m_k(n) \leftarrow m_k(n-1) + d_k(n).$$

(3) If there no error in kth differential muscle data:

- Update kth muscle data using kth differential muscle data:

$$m_k(n) \leftarrow m_k(n-1) + d_k(n).$$

side information exists or not. If side information is available, the receiver extracts the start expression (SE), the end expression (EE) and the muscle data slope (β_n). In this case the erroneous muscle data is then interpolated as

$$m_k(n) = \beta_n \times (m_k^{EE} - m_k^{SE}) + m_k^{SE} \tag{4.19}$$

where $m_k(n)$, m_k^{SE}, m_k^{EE} are the erroneous kth muscle data, the kth muscle data of the start expression, and the kth muscle data of the end expression, respectively and β_n is the muscle data slope of the nth frame.

If there is no side information, the erroneous muscle data is estimated using the following method. Since in the facial data compression considered here the transmitter sends a difference of muscle data (see Figure 4.13), the kth muscle data of the nth frame is generated as:

$$m_k(n) = m_k(n-1) + d_k(n) \tag{4.20}$$

where $d_k(n)$ represent the kth difference muscle data of the nth frame. When the receiver encounters erroneous muscle data of the nth frame, the error is corrected by setting

$$d_k(n) = d_k(n-1). \tag{4.21}$$

Since $d_k(n)$ is not necessarily the same as $d_k(n-1)$, this approach is not always effective. Nevertheless, it mitigates the propagation of distortion.

4.8 Results and discussions

In our experiment, facial animation engine (FAE) and the muscle data of six basic facial expressions are based on the FAE of the HP laboratory [53]. We made a test sequence having 1260 frames of facial animation from six basic facial expressions (i.e. happiness, anger, sadness, surprise, fear, disgust) and natural expression. The ranges of the geometrical data of facial model are (4.36, -4.36), (-8.74, 7.522) and (0.149, 9.855) in v_x, v_y and v_z, respectively. There are 42 different combinations of expression change with six basic expressions and natural expression. We assume the number of frames between two different expressions is 30. Thus, our test sequence contains all of possible combination of the expression change.

Since our research is focused on error concealment of synthetic facial animation we generated our muscle data artificially. The error concealment method proposed here is, however, equally applicable if the synthetic facial muscle data is extracted from natural video. In the encoder (Figure 4.13(a)), we consider 256 and 1024 quantization levels after differentiation each muscle data. Huffman coder is employed to compress quantized muscle data. The average bit rate of source coder are 85.5 and 81.5 bit per frame with side information and without side information, respectively.

To verify the efficiency of the proposed error concealment algorithm, we consider three scenarios: (1) transmission of compressed facial muscle data over an error-free channel, thus, there exists just quantization error, (2) transmission of compressed facial muscle data over a channel in which bit alteration error occurs and decoding is performed without error concealment, and (3) transmission of

compressed facial muscle data over the same channel as (2) but using error concealment when error occurs during transmission. We use a generalized Gilbert-Elliott channel to simulate an error prone channel. In our simulation, bit error is introduced into the compressed muscle data using Gilbert-Elliott channel error model. For our channel model, the packet size l_p is 32, the error-free interval l_s is 30. To change the error rate, we use different probability of the good state P_G and the probability of bad state P_B; Error condition 1 (EC1): $P_G = 0.995$ and $P_B = 0.0001$, and Error condition 2 (EC2): $P_G = 0.995$ and $P_B = 0.0005$. Thus, there are more errors in error condition 2 than in error condition 1.

For objective comparison, distortion between the original and the reconstructed 3-D facial animation is employed. The 3D mesh model of facial animation consists of N_T triangles, ψ_i, $i = 1, 2, \cdots, N_T$. Each triangle ψ_i has three vertices, $\psi_i = (v_{i_1}, v_{i_2}, v_{i_3})$, where $v_{i_j} = (x_{i_j}, y_{i_j}, z_{i_j}) \in \mathbb{R}^3$. If there is channel error during transmission of m_k muscle data, the receiver gets different muscle data $\bar{m}_j$ and generates different vertex data $\bar{v}_j$ as follows

$$\bar{v}_i = \begin{cases} v_i & \text{if } \alpha_{i,j} = 0 \\ \sum_{k=1,k\neq j}^{N_M} \alpha_{i,k} m_k + \alpha_{i,j} \bar{m}_j & \text{otherwise} \end{cases} \tag{4.22}$$

where coefficient $\alpha_{i,k}$ is defined in Equation (4.12). In our simulation the $\mathfrak{L}(\mathcal{S})$ is 20.8291.

Figures 4.14 and 4.15 show the PSNR representation of the facial animation. For each case, four PSNR curves are displayed based on different channel scenarios: there is no transmission error (in which case there is only quantization error),

transmission error without concealment, forward error concealment and concealment by post processing. In Figures 4.14 and 4.15 the solid line curve (—) shows the case of no transmission error where performance is affected by quantization error only. The curve with dot line (· · ·) represents the PSNR when channel error exist but there is no error concealment. The PSNR of forward error concealment and error concealment by post processing are displayed with dash-dot line curve (–·–) and dash-dot-dot line curve (–··–), respectively.

Figure 4.14 shows the PSNR of the facial animation when the number of triangles in the 3-D face model is 876. Error prone channel is modelled with error condition EC = 1. Difference muscle data are quantized with 256 and 1024 quantization levels. Figures 4.14(a), (b) and (c) represent $PSNR_{max}$, $PSNR_{avg}$ and $PSNR_{rms}$ of same rendered face, respectively, when quantization level is 256. When quantization level is 1024 the $PSNR_{max}$, $PSNR_{avg}$ and $PSNR_{rms}$ are illustrated in Figures 4.14(d), (e) and (f), respectively.

Although, Figures 4.14(a), (b) and (c) have similar error pattern curve we can see from these figures that forward error concealment and error concealment by post processing both offer little bit improvement. The quality of rendered face after error concealment by post processing can not overcome the quality of rendered face when there is only quantization error. When transmitter sends side information (I_n) error concealment by post processing improves $PSNR_{avg}$ and $PSNR_{rms}$ around 10 dB.

Figures 4.14(d)-(f) show another results when there is small quantization error. Since quantization error is small the overall quality of rendered face is better than previous set of results with 256 quantization levels. The different of error

occurred by different quantization levels is about 10 dB. In these figures we can see the effect of our proposed error concealment. In Figure 4.14(d) which represent $PSNR_{max}$, due to error in frame number 105, the $PSNR_{max}$ drops significantly. In this case, both error concealment methods perform satisfactory and PSNR improves around 4 and 6 dB by error concealment by post processing and forward error concealment, respectively.

When the channel condition is not stable the proposed error concealment algorithm give better performance. Figure 4.15 shows when there are more channel error during data transmission. We modelled channel with error condition $EC = 2$. Figures 4.15(a), (b) and (c) illustrate $PSNR_{max}$, $PSNR_{avg}$ and $PSNR_{rms}$, respectively when quantization level is 256. In these figures due to the error at frame number 13 the PSNR decreased a lot when there is no error concealment. In this case the PSNR improved about 20 dB by both error concealment methods. When error concealment is not adopted, compare to Figures 4.14(a)-(c), the error curve is much worse. But when error concealment is adapted the difference is alleviated a lot. Figures 4.15(d)-(f) show another example of an unstable channel condition. The quantization level of this simulation is 1024. Errors occurred at 13 frames over 1024 frames. Whenever error occurs the PSNR dropped significantly. Sometimes the PSNR dropped more than 10 dB (see around 200th frame). In this situation, the proposed forward error concealment and error concealment by post processing mitigate the error effectively.

From these simulation, we also find that the performance of proposed error concealment is sensitive to γ (see Equation (4.18)) and side information I_n. Small γ makes proposed error concealment method too sensitive to the change of muscle

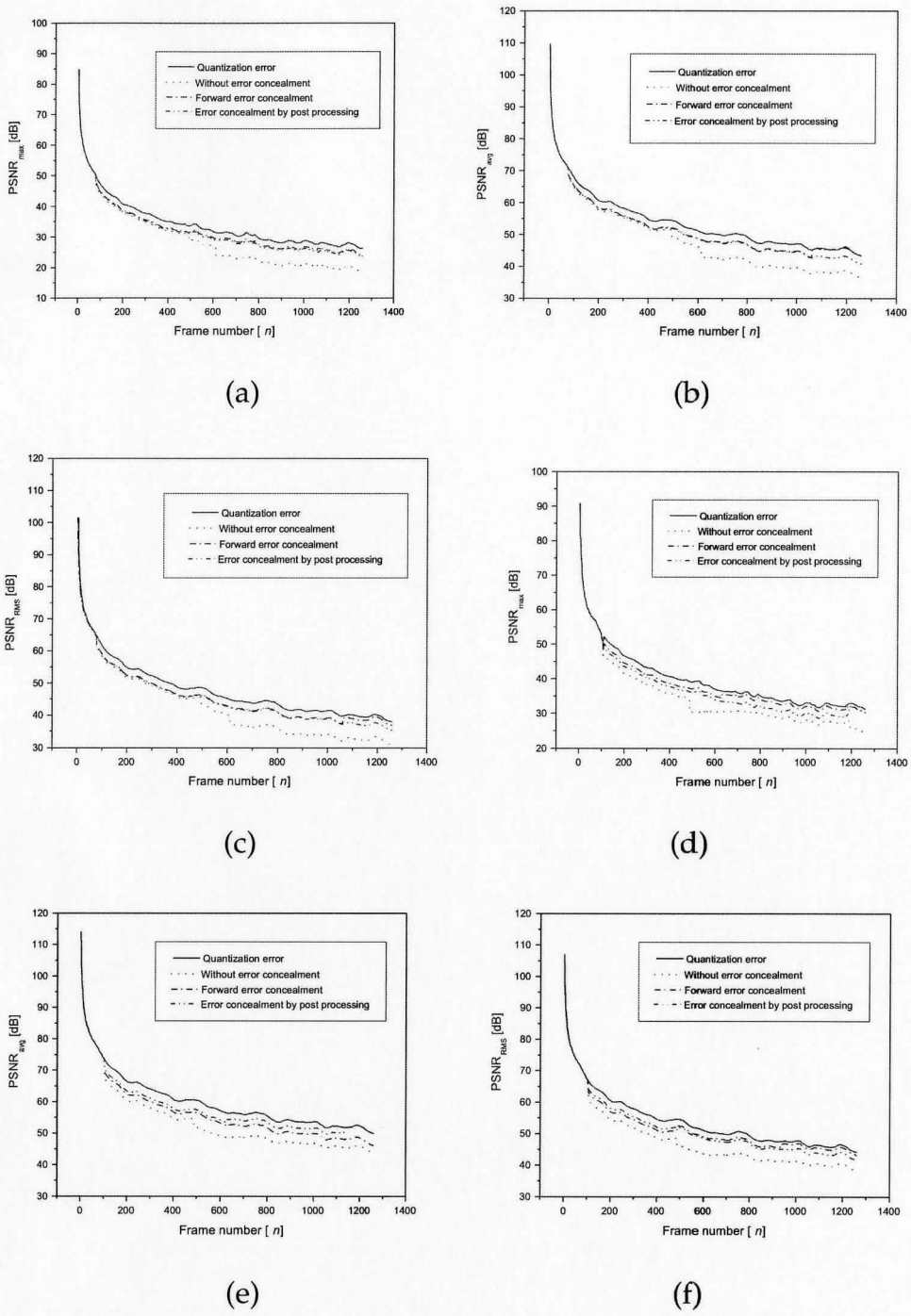


Figure 4.14: Simulation result for error condition EC = 1 and 256 quantization levels ((a),(b) and (c)), and 1024 quantization levels ((d), (e) and (f)

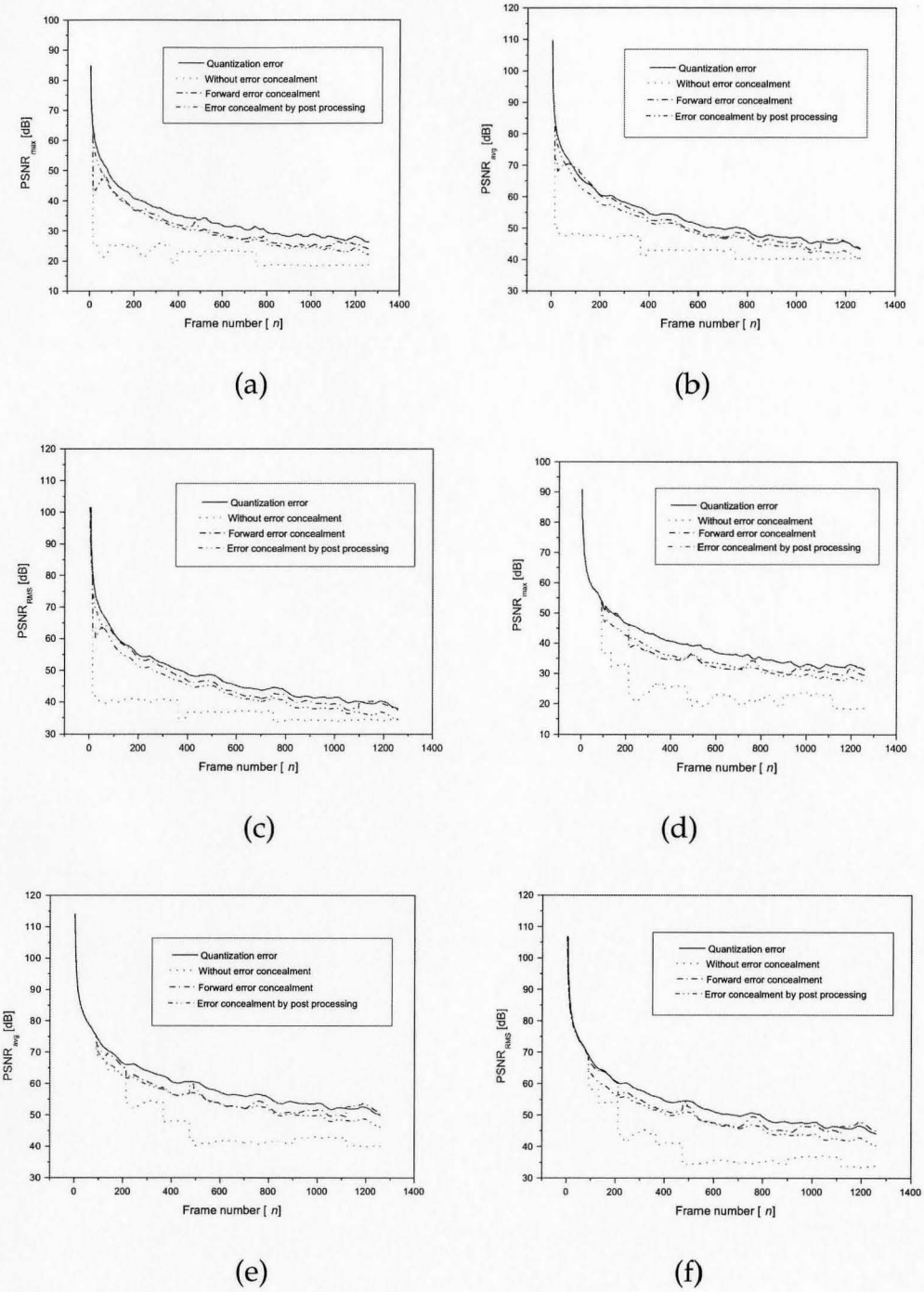


Figure 4.15: Simulation result for error condition EC = 2 and 256 quantization levels ((a),(b) and (c)), and 1024 quantization levels ((d), (e) and (f))

data. In Figures 4.15(e) and (f), for frames after 1100, the proposed error concealment with side information outperform quantization error in terms of $PSNR_{avg}$ and $PSNR_{rms}$. Thus, the performance of forward error concealment is really depend on the side information I_n. By comparing $PSNR_{avg}$ and $PSNR_{rms}$ of proposed error concealment algorithm to those without error concealment we can see that the proposed algorithm reduces the distortion of rendered facial animation. The proposed algorithm improved the distortion more than 10 dB.

Visual inspection of the final rendered 3D facial animation makes the effectiveness of the proposed error concealment algorithm even more clear. Figure 4.16 shows an example of the performance of our proposed method using 1024 quantization levels with error condition 2. Figure 4.16(a) is the last frame of the original facial animation. Figure 4.16(b) shows the typical effect of quantization error. Figure 4.16(c) shows the effect of random transmission error on the rendered face. Figures 4.16(d) and (e) shows the result of performing error concealment by post processing and forward error concealment, respectively. Note the significant distortion of the eye and mouth in Figure 4.16(c) compensated to Figure 4.16(e). Comparing Figures 4.16(d) and 4.16(e) with Figure 4.16(c) it is clear that error concealment has improved the quality of the rendered frame.

4.9 Conclusion

In this chapter, we have adapted a general MDSQ, MDVQ to 3D graphic coding. MDC is a good strategy for diversity-based network channel. The adapted MDSQ and MDVQ provides descriptions generating enough quality of reproduced 3D

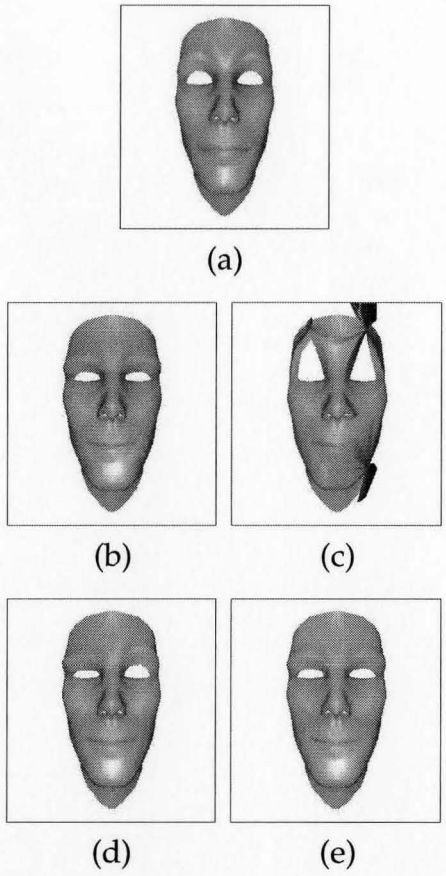


Figure 4.16: Distortion of facial animation and the performance of forward error concealment and error concealment by post processing when quantization step levels is 1024 and error condition $EC = 2$: (a) the last frame of the original facial animation, (b) the last frame with quantization error, (c) the last frame with transmission error and without error concealment, (d) the last frame with transmission error after error concealment by post processing and (e) the last frame with transmission error after forward error concealment.

graphics. Experimental results show how MDC for 3D graphics improve the performance when both channel work. Also we proposed a side decoding methods for MDC to improve the performance when channel is broken. If both descriptions are available at the decoder the decoding method is very straightforward and the performance depends on the levels of quantization. In case of a broken channel, the scenario is much different. The decoder need to reproduce input source with one description.

To reduce side distortion we proposed two side decoding algorithms. Side decoder based on diagonal element has good performance when encoder use simple IA matrix. Side decoder based on the probability of information gives better performance than other side decoders. When low bit-rate and complicated IA matrix are used the side decoder based on probability of information is recommendable. We verified the performance of each side decoder using different data sources and compared the results with the performance of central decoder. The proposed decoding method reproduce good quality of input source for both random data and 3D graphic data.

Also we have proposed an error concealment algorithm for facial animation. Our error concealment employs forward error concealment and/or error concealment by post processing. In the proposed forward error concealment, side information consisting of two basic expressions and muscle data slope is transmitted along with compressed facial data. Linear interpolation is employed to interpolate the lost data using the side information.

The two basic expressions are decided at the encoder by determining facial expression flow. For more objective comparison, we simulated with different error

rate condition. We use a simple replication of the difference muscle data in error concealment by post processing. The proposed error concealment method is simple and simulation results show that the proposed error concealment method is effective in preventing propagation of error while improving the quality of rendered facial sequences. [58]

Chapter 5

Closing remarks

This thesis focuses on communication of 3D graphic models. We provide methods for reducing the 3D data and error concealment tool to transmit over error prone channel. Only a fraction of the models are used to verify the performance of proposed idea. For efficient communication of 3D graphic models we propose area based mesh simplification. Mesh simplification reduces the size of data requiring to represent 3D graphic models. Proposed mesh simplification algorithm has good performance in terms of complexity and distortion.

Multiple descriptions for 3D graphic models are also investigated for transmission of 3D graphic data over error prone channel. Both scalar and vector quantizer are studied to generate descriptions for each 3D graphic model. Form experimental results we find that MDC is a good strategy for diversity-based network channel. We also proposed side decoding algorithm to reduce side distortion. The proposed side decoding algorithm based on probability has good performance when low bit-rate and complicated IA matrix are used.

For communication of 3D graphic model with motion via unreliable channel

forward and post-processing based error concealment methods are introduced. Facial animation model having different expressions are used to verify the performance of proposed error concealment method. Simple linear interpolation at the receiver is employed to interpolate the lost data using the side information. This thesis shows that the proposed error concealment method for moving 3D graphic models is simple and simulation results show that the proposed error concealment method is effective in preventing propagation of error while improving the quality of rendered facial sequences.

The applications of this thesis can be computer games and communication of multimedia contents. Depending on the application, the reader can use the idea of each chapter or combine those ideas together to get more desirable performance.

5.1 Future work

Since this thesis deals with a wide range of issues related to 3D graphic models the future works are also broad. Some of them are

- New distortion measure for mesh simplification
- Representing method for simplified mesh
- Designing quantizer for 3D data structure to adopt to multiple description coding
- Investigate the error concealment method with 3D models having more complicated motion
- Non-linear estimation of 3D data communication when channel error exist

Bibliography

[1] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen, "Decimation of triangle meshes," in *Computer Graphics Proc., (SIGGRAPH'92)*, pp. 65–70, July 1992.

[2] K.-L. Low and T.-S. Tan, "Model simplification using vertex-clustering," in *Proc. Interactive 3D graphics*, (Rhode Island, USA), pp. 75–87, Apr. 1997.

[3] L. Kobbelt, S. Campagna, and H.-P. Seidel, "A general framework for mesh decimation," in *Graphics Interface*, pp. 43–50, 1998.

[4] H. Hoppe, T. DeRose, and T. Duchamp, "Mesh optimization," in *ACM Computer Graphics Proc., (SIGGRAPH'93)*, Annual Conferece Series, pp. 19–26, Aug. 1993.

[5] I. Park, S. Shirani, and D. W. Capson, "Compression of 3d facial data," in *Proc. of the IEEE Canadian Conference on Electrical and Computer Engineering*, vol. 1, (Niagara fall, Canada), pp. 1151–1154, May 2004.

[6] I. Park, S. Shirani, and D. W. Capson, "Area of surface as a basis for vertex removal based mesh simplifacation," in *Proc. of the IEEE Int. Conf. in Multimedia and Expo (ICME)*, (Amsterdam, Netherlands), July 2005.

[7] I. Park, S. Shirani, and D. W. Capson, "Mesh simplification using an area based distortion measure," *Journal of Mathematical Modelling and Algorithms*, vol. 5, pp. 309–329, Feb. 2006.

[8] I. Park, S. Shirani, and D. W. Capson, "Error concealment for facial animation based on prediction of muscle data," in *Proc. of the IEEE Int. Conf. in Multimedia and Expo (ICME)*, vol. 1, (Amsterdam, Netherlands), pp. 227–280, July 2005.

[9] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: measuring error on simplified surfaces," *Computer Graphics Forum*, vol. 17, pp. 167–174, June 1998.

[10] M. Garland, *Quadric-based polygonal surface simplification*. PhD thesis, School of Computer Science Carnegie Mellon University, 500 Forbes Avenue Pittsburgh, PA 15213-3891, May 1999.

[11] N. Aspert, D. Santa-Cruz, and T. Ebrahimi, "Mesh: measuring errors between surfaces using the hausdorff distance," in *Proc. of the IEEE Int. Conf. in Multimedia and Expo (ICME)*, vol. 1, (Lausanne, Swizerland), pp. 705–708, Aug. 2002.

[12] S. Valette, A. Gouaillard, and R. Prost, "Compression of 3d triangular meshes with progressive precision," *Computers & Graphics*, vol. 28, pp. 35–42, Feb. 2004.

[13] J. Rossignac, *Handbook of discrete and computational geometry*, ch. 54 "Surface simplification and 3D geometry compression". CRS Press, second ed., 2004.

[14] A. Khodakovsky, P. Schröeder, and W. Sweldens, "Progressive geometry compression," in *Computer Graphics Proc., (SIGGRAPH 2000)*, pp. 271–278, Otc. 2000.

[15] C. Andújar, "Geometry simplification," Tech. Rep. LSI-99-2-R, Dept. of LSI, Universitat Pllitècnica de Catalunya, Diagonal 647 E-08028 Barcelona, Spain, Feb. 1999.

[16] P. S. Heckbert and M. Garland, "Survey of polygonal surface simplification algorithms," Tech. Rep. HecGar-TR-97, Carnegie-Mellon Univ., School of Computer Science, Pittsburgh, PA, May 1997.

[17] P. Cignoni, C. Montani, and R. Scopigno, "A comparison of mesh simplification algorithm," *Computers & graphics*, vol. 22, pp. 37–54, Feb. 1998.

[18] H. Hoppe, "Progressive meshes," in *ACM Computer Graphics Proc., (SIGGRAPH'96)*, Annual Conferece Series, pp. 99–108, Aug. 1996.

[19] H. Hoppe, "Efficient implementation of progressive meshes," Tech. Rep. MSR-TR-98-02, Microsoft Corporation, One Microsoft Way Redmond, WA, Jan. 1998.

[20] M. Eastlick and S. Maddock, "Triangle mesh simplification using error polyhedra," Tech. Rep. CS-01-07, Dept. of Computer Science, University of Sheffield, Portobello Road, Sheffield, UK, 1997.

[21] P. Lindstrom and G. Turk, "Fast and memory efficient polygonal simplification," in *IEEE Visualization 98 Conference Proceedings*, pp. 279–286, Otc. 1998.

[22] P. Alliez, N. Laurent, H. Sanson, and F. Schmitt, "Mesh approximation using a volume-based metric," in *Pacific Graphics 99 Conference Proceedings*, pp. 292–301, 1999.

[23] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Computer Graphics Proc., (SIGGRAPH'97)*, Annual Conference Series, 1997.

[24] L. Velho, "Mesh simplification using four-face clusters," in *IEEE Int. Conf. on Shape Modeling & amp; Applications*, (Genova, Italy), pp. 200–208, May 2001.

[25] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright, "Simplification envelops," in *Proc. SIGGRAPH'96*, pp. 119–128, Aug. 1996.

[26] V. K. Goyal, "Multiple description coding: Compression meets the network," *IEEE Signal Processing Mag.*, vol. 18, pp. 74–98, Sept. 2001.

[27] J. Wolf, A. Wyner, and J. Ziv, "Source coding for multiple descriptions," *Bell Syst. Tech. J.*, vol. 59, pp. 1417–1426, Oct. 1980.

[28] A. A. E. Gamal and T. Cover, "Achievable rates for multiple descriptions," *IEEE Trans. Inform. Theory*, vol. 28, no. 6, pp. 851–857, 1993.

[29] I. Radulovic and P. Ffossard, "Fast index assignment for balanced n-description scalar quantization," in *Proc. IEEE Int. Conf. Data Compression (DCC 2005)*, 2005.

[30] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Trans. Inform. Theory*, vol. 44, pp. 2325–2383, Oct. 1998.

[31] J. Cardinal, "Entorpy constrained index assignment for multiple description quanitzers," *IEEE Trans. Signal Processing*, vol. 52, pp. 265–270, Jan. 2004.

[32] V. Vaishampayan, "Design of multiple description scalar quantizers," *IEEE Trans. Inform. Theory*, vol. 39, pp. 821–834, May 1993.

[33] K. Sayood, *Introduction to Data Compression*. San Diego, CA: Morgan Kaufmann, 2nd ed., 2000.

[34] T. Guionnet, C. Guillemot, and S. Pateux, "Embedidded multiple description coding for progressive image transmission over unreliable channels," in *Proc. IEEE Int. Conf. on Image Proc.(ICIP'2001)*, vol. 1, (Thessaloniki, Greece), pp. 94–97, Oct. 2001.

[35] M. Wu, A. Vetro, and C. W. Chen, "Multiple description image coding with distributed source coding and side information," *SPIE Multimedia Systems and Applications VII*, vol. 5600, pp. 120–127, Oct. 2004.

[36] S. D. Servetto, K. Ramchandran, V. A. Vaishampayan, and K. Nahrstedt, "Multiple description wavelet based image coding," *IEEE Trans. Image Processing*, vol. 9, pp. 813–1821, May 2000.

[37] C. Tian and S. S. Hemami, "A new class of multiple description scalar quantizer and its application to image coding," *IEEE Signal Process. Let.*, vol. 12, pp. 329–332, Apr. 2005.

[38] Y. C. L. adn Y. Altunbasak and R. M. Mersereau, "Coordinated application of multiple description scalar qunatization and error concealment for error-resilient mpeg video streaming," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, pp. 457–468, Apr. 2005.

[39] I. V. Bajic and J. W. Woods, "Domain based multiple description coding of images and video," *IEEE Trans. Image Processing*, vol. 12, pp. 1211–1225, Oct. 2003.

[40] V. K. Goyal, J. A. Kelner, and J. Kovačević, "Multiple description vector quantization with a coarse lattice," *IEEE Trans. Inform. Theory*, vol. 48, pp. 781–788, March 2002.

[41] I. 14496-2, "Information technology- generic coding of audio-visual object, part 2: Visual (final draft of international standard)," 1998.

[42] I. JTC1/SC29/WG11, "Overview of the mpeg-4 standard, n2725," Mar. 1999.

[43] K. Waters, "A muscle model for animating three-dimensional facial expressions," *Comput. Graph.*, vol. 21, no. 4, pp. 17–24, 1987.

[44] F. Lavagetto and R. Pockaj, "An efficient use of mpeg-4 fap interpolation for facial animation at 70 bits/frame," *IEEE Trans. Circuit Syst. Video Technol.*, vol. CSVT-11, pp. 1085–1097, Oct. 2001.

[45] H. T. *et al.*, "Compression of mpeg-4 facial animation parameters for transmission of talking heads," *IEEE Trans. Circuit Syst. Video Technol.*, vol. CSVT-9, pp. 264–276, Mar. 1999.

[46] J. Ahlberg and H. Li, "Representing and compression facial animation parameters using facial action basis function," *IEEE Trans. Circuit Syst. Video Technol.*, vol. CSVT-9, pp. 405–410, Apr. 1999.

[47] S.Varakliotis, S. Hailes, and J. Osterman, "Repair option for 3-d wireframe model animation sequences," in *Proc. of the IEEE Int. Conf. in Multimedia and Expo (ICME)*, vol. 1, (Lausanne, Swizerland), pp. 137–140, Aug. 2002.

[48] T. Ishikawa, H. Sera, S. Morishima, and D. Terzopoulos, "Facial image reconstruction by estiamted muscle parameter," in *Proc. 3rd IEEE Int. Conf. Automatic Face and Gesture Recognition*, (Nara, Japan), pp. 342–347, Apr. 1998.

[49] I. Essa and A. Pentland, "Coding, analysis, interpolation, and recognition of facial expressions," *IEEE Trans. Pattern Analysis and Machine Inteligence*, vol. PAMI-19, pp. 757–763, July 1997.

[50] G. D. *et al.*, "Classifying facial actions," *IEEE Trans. Pattern Analysis and Machine Inteligence*, vol. PAMI-21, pp. 974–989, Oct. 1999.

[51] H. Ohta, H. Saji, and H. Nakatani, "Recognition of facial expressions unsing muscle-based feature moldels," in *Proc. 14th Int. Conf. Pattern Recognition*, vol. 2, (Brisbane, Australia), pp. 1379–1381, Aug. 1998.

[52] Y. Lee, D. Terzopoulos, and K. Waters, "Construction physics-based facial models of individuals," in *Proc. Graphics Interface*, (Toronto, ON, Canada), pp. 1–8, May 1993.

[53] C. research laboratory of HP [online], "Available: http://crl.research.compaq.com/projects/facial/facial.html."

[54] G. A. Regib, Y. Altunbasak, and J. Rossignac, "Error-resilient transmission of 3d models." to appear in ACM Transactions on Graphics, Apr. 2005.

[55] M. Mordechai and B.-D. Israel, "Capacity and coding for the gilbert-elliott channels," *IEEE Trans. Inform. Theory*, vol. 35, pp. 1277–1290, Nov. 1989.

[56] B. Wong and C. Leung, "On computing undetected error probabilities on the gilbert channel," *IEEE Trans. Commun.*, vol. 43, pp. 2657–2661, Nov. 1995.

[57] Y. Wang and Q. F. Zhu, "Error control and concealment for video communication: a review," in *Proc. IEEE*, vol. 86, pp. 974–997, May 1998.

[58] D. Santa-Cruz and T. Ebrahimi, "Coding of 3d virtual objects with nurbs," *Signal Processing*, vol. 82, no. 11, pp. 1581–1593, 2002.