RICE UNIVERSITY

3D Geometric Coding using Mixture Models and the Estimation Quantization Algorithm

by

Sridhar Lavu

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE Master of Science

APPROVED, THESIS COMMITTEE:

Richard Baraniuk, Chair
Professor of Electrical and Computer Engineering

Hyeokho Choi
Faculty Fellow in Electrical and Computer Engineering

Michael Orchard
Professor of Electrical and Computer Engineering

Joe Warren
Professor of Computer Science

Houston, Texas

September, 2002
3D Geometry Coding using Mixture Models and the Estimation Quantization Algorithm

Sridhar Lavu

Abstract

3D surfaces are used in applications such as animations, 3D object modeling and visualization. The geometries of such surfaces are often approximated using polygonal meshes. This thesis aims to compress 3D geometry meshes by using an algorithm based on normal meshes and the Estimation-Quantization (EQ) algorithm. Normal meshes are multilevel representations where finer level vertices lie in a direction normal to the local surface and therefore compress the vertex data to one scalar value per vertex. A mixture distribution model is used for the wavelet coefficients. The EQ algorithm uses the local neighborhood information and Rate-Distortion optimization to encode the wavelet coefficients. We achieve performance gains of 0.5-1dB compared to the zerotree coder for normal meshes.
Acknowledgments

My sincere thanks to Dr. Richard Baraniuk and Dr. Hyeokho Choi for their valuable insights and guidance and the research support at Rice University; to Dr. Michael Orchard and Dr. Joe Warren for being part of my thesis committee and for their invaluable feedback; to Dr. Andrei Khodakovsky and Dr. Igor Guskov for the normal mesh data sets; to my friends especially Farbod, Michael and Bahar for their constant support and friendship; and last but not the least, to my mother for all her encouragement and support.
# Contents

Abstract ........................................... ii  
Acknowledgments .................................... iii  
List of Tables ..................................... viii  
List of Illustrations ............................... ix  

1 Introduction .................................... 1  
1.1 3D geometry surfaces .......................... 1  
1.2 Contribution of thesis .......................... 1  
1.3 Organization of thesis .......................... 2  

2 3D Surface Data and Normal Meshes .......... 4  
2.1 3D geometry surfaces .......................... 4  
2.2 Remeshing ...................................... 7  
2.2.1 Mesh simplification .......................... 8  
2.2.2 Subdivision .................................. 8  
2.3 Normal meshes .................................. 11  
2.4 Wavelet transforms for meshes ............... 15  
2.4.1 Lifting ...................................... 16  
2.4.2 Wavelet transforms for normal meshes ....... 17  
2.5 Related work: zerotree coder for normal meshes .... 18  
2.6 Related work: EQ coder for 2D images .......... 22
3 EQ Coder for Normal Meshes

3.1 Mixture models for the wavelet coefficients ........................................... 23
3.2 Estimate step ................................................................................. 24
   3.2.1 Estimating the shape parameter .................................................. 24
   3.2.2 Estimating the standard deviation parameter ............................... 25
3.3 Quantize step ............................................................................. 26
   3.3.1 Rate-Distortion optimization ....................................................... 27
   3.3.2 Rate-Distortion curves ............................................................... 28
   3.3.3 Entropy coding .......................................................................... 29
3.4 Unpredictable set ......................................................................... 30
   3.4.1 Characterizing the unpredictable set ........................................... 30
   3.4.2 Initial partitioning of the coefficients ........................................ 31
   3.4.3 Tangential wavelet coefficients .................................................. 32
3.5 Vertex neighborhood ................................................................. 32
3.6 Vertex ordering ........................................................................... 33
   3.6.1 Vertex ordering inside a base triangle ....................................... 34
   3.6.2 Base triangle ordering ............................................................. 35
   3.6.3 Base triangle orientation .......................................................... 35
3.7 Scaling coefficients ................................................................. 37
3.8 Mesh connectivity ..................................................................... 38

4 Algorithm Implementation Details .................................................. 40

4.1 Mesh file format ................................................................. 40
4.2 Encoding mesh connectivity .................................................... 40
   4.2.1 TG coder correction ............................................................... 40
   4.2.2 Array representation for mesh connectivity ............................... 41
5 Results and Analysis

5.1 Error metrics .................................................. 51
5.2 Error tools .................................................... 53
5.3 Data sets used ............................................... 54
5.4 Error measurements ......................................... 55
5.5 Results ......................................................... 56
5.5.1 Performance of the EQ coder ........................... 56
5.5.2 Comparison of the EQ and the zerotree coders .......... 61
5.5.3 Comparison of different wavelet transforms ............ 61
5.5.4 Reconstructed meshes at different bit-rates ........... 71
5.5.5 Progressive nature of the coders ....................... 71
6 Discussion and Conclusions 75
  6.1 Conclusions ........................................ 75
  6.2 Future work ........................................ 75

Bibliography 77

A Asymmetric Hausdorff Distance 81

B Metro and MeshDev Sample Output 82
  B.1 Metro sample output .................................. 82
  B.2 MeshDev sample output .............................. 84
Tables

5.1 The Venus head, the horse and the rabbit normal meshes. ........ 55
5.2 Venus head mesh PSNR comparison. ............................ 70
5.3 Horse mesh PSNR comparison. ................................. 70
5.4 Rabbit mesh PSNR comparison. ................................. 70
Illustrations

2.1 A simple pyramid mesh. .................................................. 5
2.2 The Venus head mesh. ..................................................... 5
2.3 Mesh geometry and mesh connectivity of a simple pyramid mesh. . 6
2.4 Linear interpolating subdivision example. ........................... 8
2.5 Masks for the Loop subdivision scheme. ............................ 10
2.6 Masks for the modified butterfly subdivision scheme. ............. 10
2.7 Multilevel normal mesh representation of the rabbit mesh. ....... 12
2.8 Finding a normal vertex. .................................................. 13
2.9 Normal mesh representation for the Venus head model. .......... 14
2.10 Lifting scheme: forward transform ................................... 16
2.11 Lifting scheme: inverse transform ................................. 17
2.12 Lifted Loop wavelet transform reconstruction filters. ........... 18
2.13 Histogram plots of normal wavelet coefficients from the horse normal mesh starting with the base level. ....................... 19
2.14 Histogram plots of tangential wavelet coefficients from the horse normal mesh starting with the base level. ................... 20
2.15 Edge based quadtree for meshes. .................................... 21
2.16 Quadtrees inside a base triangle. .................................... 21

3.1 Simple $3 \times 3$ pixel neighborhood in an image. .................. 33
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2</td>
<td>Simple four-neighbor rule for a regular vertex in a mesh.</td>
<td>33</td>
</tr>
<tr>
<td>3.3</td>
<td>Neighborhood of a vertex with parent edge connected by regular</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>vertices.</td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>Neighborhood of a vertex with parent edge connected by</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>extraordinary vertices.</td>
<td></td>
</tr>
<tr>
<td>3.5</td>
<td>Raster-scanning and causal neighborhoods for image pixels.</td>
<td>35</td>
</tr>
<tr>
<td>3.6</td>
<td>Vertex ordering inside base triangles.</td>
<td>36</td>
</tr>
<tr>
<td>3.7</td>
<td>Base triangle ordering.</td>
<td>37</td>
</tr>
<tr>
<td>3.8</td>
<td>Causal neighborhood using a simple four-neighbor rule.</td>
<td>38</td>
</tr>
<tr>
<td>4.1</td>
<td>Array representation of a base triangle.</td>
<td>42</td>
</tr>
<tr>
<td>4.2</td>
<td>GGD fit for the <em>unpredictable</em> tangential coefficients.</td>
<td>47</td>
</tr>
<tr>
<td>4.3</td>
<td>GGD fit for the <em>unpredictable</em> normal coefficients.</td>
<td>48</td>
</tr>
<tr>
<td>4.4</td>
<td>GGD fit for the <em>predictable</em> normal coefficients.</td>
<td>49</td>
</tr>
<tr>
<td>5.1</td>
<td>The Venus head, the horse and the rabbit models.</td>
<td>54</td>
</tr>
<tr>
<td>5.2</td>
<td>Metro error using the EQ coder for the Venus head, the horse and the</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>rabbit models.</td>
<td></td>
</tr>
<tr>
<td>5.3</td>
<td>Metro PSNR in dB using the EQ coder for the Venus head, the horse</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>and the rabbit models.</td>
<td></td>
</tr>
<tr>
<td>5.4</td>
<td>MeshDev error using the EQ coder for the Venus head, the horse and</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>the rabbit models.</td>
<td></td>
</tr>
<tr>
<td>5.5</td>
<td>GGD fit for the normal coefficients of the Venus head normal mesh</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>normalized by the standard deviation of the complete neighborhood.</td>
<td></td>
</tr>
<tr>
<td>5.6</td>
<td>Bounds for the EQ coder (the Venus head normal mesh) in terms of</td>
<td>62</td>
</tr>
<tr>
<td></td>
<td>the Metro error, the Metro PSNR and the MeshDev error.</td>
<td></td>
</tr>
</tbody>
</table>
5.7 Comparing the EQ and zerotree coders using Metro error, Metro
PSNR and MeshDev error for the Venus head normal mesh. ........... 63
5.8 Comparing the EQ and zerotree coders using Metro error, Metro
PSNR and MeshDev error for the horse normal mesh. ................. 64
5.9 Comparing the EQ and zerotree coders using Metro error, Metro
PSNR and MeshDev error for the rabbit normal mesh. ............... 65
5.10 Comparing the EQ and zerotree coders using the Loop and butterfly
wavelet transforms. (a) Histograms of Loop and unlifted butterfly
wavelet coefficients for the Venus head normal mesh at scale 4.
(b)-(d) Metro error, Metro PSNR and MeshDev error for the Venus
head normal mesh. .............................................. 67
5.11 Comparing the different wavelet transforms for the EQ and zerotree
coders. The Venus head normal mesh is used for these plots. ....... 68
5.12 Comparing the EQ and zerotree coders using the lifted butterfly
wavelet transform for the horse mesh. ............................ 69
5.13 Comparing the EQ and zerotree coders using the lifted butterfly
wavelet transform for the rabbit mesh. ............................ 69
5.14 The Venus head normal mesh reconstruction using EQ coder at
different bits-per-vertex (bpv). ................................. 72
5.15 Comparing the progressive nature of the EQ and zerotree coders
using the Metro error and PSNR for the Venus head mesh. EQ 1, EQ
2 and EQ 3 correspond to $\lambda = 10^{-4}$, $\lambda = 10^{-6}$ and $\lambda = 10^{-8}$
respectively, where $\lambda$ is the rate distortion parameter used for
encoding the mesh with the EQ coder. .............................. 73

A.1 Asymmetric nature of Hausdorff distance. ......................... 81
Chapter 1

Introduction

1.1 3D geometry surfaces

3-dimensional (3D) graphics applications such as animations, object modeling, and visualization applications use complex geometry models to describe 3D surfaces. These surfaces contain geometry, color, texture, and other information. The geometries of such surfaces are often represented using polygonal meshes. The typical size of a polygonal mesh is, however, very large making it difficult to edit, store, and render the meshes. Furthermore, the rapid development of the Internet has increased the use of these graphics applications and thus has increased the need to develop good compression schemes for the 3D geometry models.

1.2 Contribution of thesis

This thesis proposes a new algorithm to compress the geometry of 3D surfaces. It is based on normal meshes and the Estimation-Quantization (EQ) algorithm. Normal meshes are efficient multilevel representations where each vertex is represented using a single scalar offset in a direction normal to the local surface. We assume that the wavelet coefficients of a normal mesh are locally correlated and are modeled using a mixture model distribution whose parameters are estimated from the local vertex neighborhoods. We use the EQ algorithm to estimate the parameters and encode the
wavelet coefficients using Rate-Distortion (RD) optimization. An entropy coder is used to encode the quantized symbols from the EQ coder.

1.3 Organization of thesis

In Chapter 2, we discuss 3D surface geometry data and their multilevel representations. In particular, we describe normal mesh representations and wavelet transforms for meshes using subdivision and the lifting schemes. The butterfly wavelet transform is applied to normal meshes and the distribution of the coefficients is given. Compression of the wavelet coefficients using zerotrees is briefly explained. The EQ coder is introduced and the application of the EQ coder to 2D images is discussed.

In Chapter 3, the EQ coder is extended to the normal mesh case. The wavelet coefficients are modeled using a generalized Gaussian distribution (GGD). The estimation and the quantization steps of the EQ algorithm are explained. The estimation step estimates the parameters of the GGD model and quantization step quantizes the coefficients using R-D optimization. Encoding methods for the wavelet coefficients, the scaling coefficients and the connectivity information are given.

In Chapter 4, implementation details of the EQ algorithm are given. The estimation and the quantization steps are discussed in greater detail. Finally, the decoder algorithm is discussed.

In Chapter 5, the error measures and the performance results of the EQ coder are given. In particular, error measures based on the Hausdorff distance are discussed.
The performance of the EQ coder is compared with the zerotree coder for normal meshes. We also give bounds for the performance of the EQ coder and compare the effect of using different wavelet transforms.

In Chapter 6, we briefly discuss the performance of the EQ coder for normal meshes followed by conclusions and directions for future work.
Chapter 2

3D Surface Data and Normal Meshes

2.1 3D geometry surfaces

3D surfaces can be used in applications such as computer graphics, animations [1], virtual reality, medical and visualization applications, 3D object modeling and design applications. These surfaces often consist of different data such as color, texture and geometry. The 3D geometry surfaces contain geometry information alone with no color or texture or any other information.

The geometries of 3D surfaces are represented using polygonal meshes. Figure 2.1 shows a simple 3D pyramid. A polygonal mesh consisting of four triangles and one quadrilateral is used to describe the simple pyramid surface. Figure 2.2 shows a more complex mesh used to describe the Venus head with around a 100,000 triangles.

3D scanners [2] are one way of generating 3D surfaces. They scan objects from different angles using very fine resolutions. The points obtained from 3D scanning are assembled together to form a surface of point clouds. These point clouds are converted to form a polygonal mesh approximation to the surface [3]. Thus the geometries of 3D surfaces are approximated using polygonal meshes.

3D polygonal meshes often contain large numbers of polygons resulting in huge data sizes. The mesh data is split into two parts, the mesh geometry and mesh con-
connectivity. The mesh geometry part consists of the list of vertices with position or coordinate information \((x, y, z)\) for each vertex. Three floating point numbers are used for each vertex. The mesh connectivity part consists of the list of polygons with each polygon represented using a set of indices corresponding to the vertices. Figure 2.3 shows the mesh geometry and the mesh connectivity for the simple pyramid mesh shown in figure 2.1. Note that both mesh geometry and mesh connectivity are necessary to represent a mesh as shown in figure 2.3.

A 3D surface is denoted as \(S\) and the mesh that describes the surface \(S\) is denoted as \(M\). The mesh \(M\) is represented using \((P, K)\), where \(P\) is the mesh geometry and \(K\) is the mesh connectivity. The mesh geometry lists all the vertices with coordinate information \(P = \{p_i = (x_i, y_i, z_i) \in \mathbb{R}^3 \mid 1 \leq i \leq N\}\) where \(N\) is the number of vertices in the mesh. The mesh connectivity lists all the polygons in the mesh, for
example, a triangular face is represented as \( \{ i, j, k \} \). Several file formats such as the Open Inventor format [4], the Virtual Reality Modeling Language (VRML), the Simple Model File format (SMF) [5] and the Caltech DAT format [6] are used to represent the mesh data.

3D meshes contain a large number of polygons dealing with connectivity and a large number of vertices which includes three coordinates per vertex. Therefore, they require much more compression than 2D images. A wavelet transform of a normal mesh multilevel representation gives a sparse representation for the mesh geometry. We propose a compression technique for the normal mesh wavelet coefficients that is based on the EQ algorithm. In Section 2.2, we discuss multilevel representations obtained using mesh simplification and subdivision schemes.
2.2 Remeshing

The original mesh usually has a large number of polygons and vertices. Multilevel representations using a base mesh and a sequence of successive larger difference meshes provide with an efficient representation for meshes. Mesh simplification and subdivision are used to obtain a multilevel representation of the original mesh. Subdivision connectivity is necessary for meshes in order to define wavelet transforms.

We also want to convert the original irregular mesh into a semi-regular or a regular mesh. A mesh is said to be regular, semi-regular or irregular based on the valence of the vertices in the mesh. The valence of a vertex is the number of vertices that are directly connected to the given vertex. A vertex with a valence of 6 is called a regular vertex and a vertex with a valence different from 6 is called an extraordinary vertex. A regular mesh is a mesh with all regular vertices, a semi-regular mesh is a mesh where only a small number of vertices are extraordinary vertices and an irregular mesh is a mesh with many extraordinary vertices. Encoding mesh connectivity and editing meshes is much simpler for regular or semi-regular meshes compared to the irregular meshes.

Original or initial meshes are often irregular meshes and are converted to a regular or a semi-regular mesh with subdivision connectivity. This conversion process is referred to as remeshing. In the following sections, we discuss mesh simplification and subdivision and explain how they are used for remeshing in order to obtain multilevel representations such as the normal meshes.
2.2.1 Mesh simplification

Mesh simplification is the process of reducing the number of polygons and vertices in the original mesh. Mesh simplification usually involves application of techniques such as vertex removal and polygon merging. Removing vertices and merging polygons are some of the ways of mesh simplification. An example of a mesh simplification algorithm is the Garland-Heckbert simplification which is based on merging vertex pairs and the Quadric Error metric [7].

2.2.2 Subdivision

Subdivision [8] is the process of obtaining a smooth surface $S$ as a limit of a sequence of successive refinements $F$, starting with a base mesh $M_0$. Subdivision schemes are similar to splines [9] but are more general than splines because subdivision schemes can be defined for functions with arbitrary topology. The iterative transform $F$ is used to obtain a finer level mesh representation $M_{j+1}$ of the surface $S$ from a coarse level mesh $M_j$ and is expressed as $M_{j+1} = F(M_j)$. Figure 2.4 shows an example of subdivision where the linear interpolating subdivision scheme is used for a simple pyramid mesh.

![Figure 2.4](image)

*Figure 2.4* Linear interpolating subdivision example.
More complex subdivision rules are often used to obtain a smooth surface starting from a base mesh. These rules are based on principles such as compact support, efficient algorithms and affine invariance. Subdivision schemes can be classified into two types, interpolating subdivision and approximating subdivision. In approximating subdivision, the vertices at the current level are updated after the new vertices are added at each new level, whereas in interpolating subdivision, the existing vertices do not change as we introduce new vertices at each new level. The new vertices introduced at each step are usually referred to as the odd vertices and the old vertices are usually referred to as the even vertices. The Loop subdivision scheme [10] and the butterfly subdivision scheme [11] are examples of approximating and interpolating subdivision schemes respectively. The modified butterfly subdivision scheme [12] is an improvement over the original butterfly scheme. Figure 2.5 and figure 2.6 show the rules used for the Loop and the modified butterfly subdivision schemes respectively.

Figure 2.5 shows the masks used in the Loop subdivision scheme for the interior and the boundary vertices. The figure includes the masks for the even vertices since the Loop subdivision scheme is an approximating subdivision scheme.

Figure 2.6 shows the masks for the odd vertices in the modified butterfly subdivision scheme. Each odd vertex is associated with a parent edge. The masks are different depending on whether the vertices at the ends of the parent edge are extraordinary vertices, regular vertices or boundary vertices. If both the vertices at the ends of the parent edge have extraordinary valence, then an average of the two results obtained using the last mask shown in figure 2.6 (d) is used. The weights $s_i$ for $i = 0 \ldots n - 1$ in the last mask are given in [12]. The butterfly subdivision scheme is an interpolating subdivision scheme and therefore the even vertices are fixed at finer
levels.

Figure 2.5  Masks for the Loop subdivision scheme.

Figure 2.6  Masks for the modified butterfly subdivision scheme.
Combining mesh simplification and subdivision

We can now combine mesh simplification and subdivision to obtain a multilevel representation with subdivision connectivity. Mesh simplification is used to obtain a base mesh from the original irregular mesh. The base mesh usually contains very few triangles and vertices compared to the original mesh. Subdivision of the base mesh is used to obtain the mesh at the next finer level and this process is iterated until a mesh with the desired resolution is obtained. The difference vector between the vertices predicted by subdivision and the actual surface is stored for each new vertex instead of the actual coordinates of the new vertex. Normal remeshing [13], MAPS remeshing [14] are examples of remeshing techniques which convert the original irregular meshes into multilevel meshes with semi-regular connectivity. Figure 2.7 shows the rabbit mesh at different levels of resolution obtained using normal remeshing. The starting base mesh contains 71 vertices and 138 triangles and the final mesh contains 70658 vertices and 141312 triangles. In the next section, normal remeshing is discussed in further detail.

2.3 Normal meshes

Normal meshes [13] are semi-regular multilevel meshes with subdivision connectivity where the finer level vertices lie in a normal direction defined by the local vertex neighborhood at the coarser level. Each new vertex is found by first predicting the base point and then drawing a line through this base point in a direction that is normal to the local surface to find the intersection with the original surface. Subdivision schemes are used to predict the base point and the normal direction. The process of converting the original mesh to a regular or a semi-regular normal mesh is referred
Figure 2.7 Multilevel normal mesh representation of the rabbit mesh.
to as normal remeshing.

In order to obtain a normal mesh from the original mesh, the Garland-Heckbert mesh simplification algorithm [7] is used to obtain an approximate base mesh. The modified butterfly subdivision scheme [12] is then used to predict the base point $b$ and the normal direction $\vec{n}$ for each new vertex. Figure 2.8 shows the normal line $\vec{n}$ passing through base point $b$. The intersection of this normal line with the original surface is used to find the new vertex associated with the parent edge $(s_1, s_2)$ in the figure.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure2_8.png}
\caption{Finding a normal vertex.}
\end{figure}

If there are no intersection points for the normal line or if the intersection point is too far from the base point $b$ in the parametric domain, then a different scheme is used to find the new non-normal vertex. The number of non-normal vertices is typically very small compared to the total number of vertices in the mesh.

Figure 2.9 shows a normal mesh representation of the Venus head model. Figure 2.9 (a) shows the base mesh that is used to construct the normal mesh and the (b) shows the mesh after one iteration of subdivision with the dotted lines showing the
triangles of the base mesh. (c) shows the mesh after three levels of subdivision and (d) shows the final mesh after six levels of subdivision. The base mesh contains 42 vertices and 80 triangles and the final mesh contains 163842 vertices and 327680 triangles.

Figure 2.9 Normal mesh representation for the Venus head model.
The local coordinates are used to describe the vector offsets between the new vertex and the predicted vertex using one normal component and two tangential components for each new vertex. The tangential components of the offsets are usually zero as normal meshes have a majority of the new vertices in the normal direction. Thus, each new vertex is expressed using a single scalar value rather than a vector of three coefficients. Normal meshes exploit the correlation between the three coefficients of the vector offset and therefore obtain an efficient representation for the mesh. A wavelet transform is used to further decorrelate the normal components and the tangential components of the offsets for the new vertices. In the next section, the wavelet transforms for normal meshes are discussed.

2.4 Wavelet transforms for meshes

The original meshes often describe smooth surfaces and therefore the offsets for the new vertices exhibit local correlation. A wavelet transform is used to exploit this correlation.

The traditional first generation wavelet transforms are usually defined for regular data sets using translation and dilation operations of a fixed function. These operations become simple algebraic operations in the Fourier domain making it easy to construct wavelet transforms in the Fourier domain. However, since the mesh data is an irregular data set, the Fourier domain cannot be used to define wavelet transforms for meshes. Instead, the lifting scheme [15] is used to extend the wavelet transform definitions for irregular data sets such as the mesh data.
2.4.1 Lifting

The lifting scheme [15] consists of three steps - the split, the predict and the update steps. The split step partitions the data into two disjoint sets - the odd subset and even subset. The even subset is used to predict the odd subset. The prediction errors are usually small since most real-life data exhibit local correlation. The prediction errors are used to update the even set in the update step. The update step is usually chosen in such a way that the resulting coefficients have some nice mathematical property such as vanishing moments, the same DC component as that of the original signal.

Figure 2.10 shows the lifting steps for the data $x[n]$. The data $x[n]$ is split into an even subset $x_e[n] = x[2n]$ and an odd subsets $x_o[n] = x[2n + 1]$. The even subset is used to predict the odd subset and the prediction errors $x_o[n] - P(x_e[n])$ are stored as $d[n]$. The prediction errors are used to update the even subset $x_e[n] + U(d[n])$ and the updated coefficients are stored as $c[n]$. Here $d[n]$ and $c[n]$ correspond to wavelet and scaling coefficients respectively. The inverse wavelet transform is obtained by inverting the lifting steps. Figure 2.11 shows the inverse lifting transform.

![Lifting scheme: forward transform](image)
The lifting scheme is used to define wavelet transforms for meshes. Subdivision can be used as the predict step in lifting. The update step can be omitted or can be chosen based on some nice mathematical property such as vanishing moments. The Loop wavelet transform [16] is defined using the Loop subdivision scheme [10] shown in figure 2.5. Figure 2.12 shows the low pass and the high pass reconstruction filters for the Loop wavelet transform. These are filters for the regular vertices. The unlifted butterfly wavelet transform is defined using the modified butterfly subdivision scheme [12] as the predict step. It has no update step. The lifted butterfly wavelet transform is defined using the same subdivision scheme and an update step is proposed in [17].

2.4.2 Wavelet transforms for normal meshes

The wavelet coefficients from normal meshes are expressed using the local coordinate system. Since the normal meshes contain vertices with normal offsets, the tangential components of a majority of the wavelet coefficients are zero if the wavelet transform and the subdivision scheme used to construct the normal meshes are the same. Some of the tangential components are nonzero, as some of the vertices in the normal mesh are non-normal. Figure 2.13 and figure 2.14 illustrate the distribution of the coeffi-
Figure 2.12 Lifted Loop wavelet transform reconstruction filters.

The wavelet coefficients from normal meshes are encoded using a zerotree based coder [18, 19]. The zerotree algorithm for normal meshes is discussed in the next section.

2.5 Related work: zerotree coder for normal meshes

Normal meshes have semi-regular connectivity allowing us to construct quadtrees based on edges. Figure 2.15 shows a parent edge $e_{j,k}$ with four child edges $e_{j+1,4k+1:4k+4}$. Each edge of a base triangle (a triangle in the base mesh) forms the root of a quadtree. The edge based quadtree is used to construct a quadtree of vertices, where each edge is associated with a vertex.
Figure 2.13  Histogram plots of normal wavelet coefficients from the horse normal mesh starting with the base level.
Figure 2.14  Histogram plots of tangential wavelet coefficients from the horse normal mesh starting with the base level.
The zerotree coder [18, 19] uses three separate zero trees for the normal and the two tangential components of the wavelet coefficients. The vertices from the base mesh (base vertices) are encoded separately using uniform quantization and the bit planes are interleaved with the bit planes from the zerotrees. The TG coder for meshes [20] is used to encode the base mesh connectivity. The mesh connectivity at finer levels is embedded within the zerotrees for each base edge.

The zerotrees of normal meshes use edge based quadtrees. Figure 2.16 shows a base triangle with each of its edges forming the root of a quadtree. Note that the vertices inside the base triangle are encoded using three separate zerotrees. Therefore, the zerotree structure may not completely exploit the correlation between these vertices.
We propose that encoding all the vertices inside a base triangle together will give a better performance compared to the zerotree coder. We propose to compress the wavelet coefficients from normal meshes using the EQ algorithm [21] as it uses local context information to encode the wavelet coefficients. The EQ based compression algorithm for meshes is introduced in Chapter 3. The EQ algorithm was originally used for 2D image coding in [21]. In the next section, we briefly discuss the EQ algorithm for 2D images.

2.6 Related work: EQ coder for 2D images

The EQ coder for 2D images was proposed in [21] and is based on an Estimation-Quantization (EQ) framework. The wavelets coefficients of the 2D image are modeled using an independent zero-mean Generalized Gaussian distribution (GGD) model [22] with a fixed unknown shape parameter for each subband and unknown slowly spatially-varying variances. The estimate of the variance is obtained from the causal quantized spatial neighborhoods. The wavelet coefficients are quantized and encoded based on the estimated parameters for the GGD model. The choice of the quantizer is based on an off-line Rate-Distortion (R-D) optimization. The quantized symbols are entropy-encoded using an arithmetic coder [23]. In the next Chapter, we extend the EQ based compression algorithm to normal meshes.
Chapter 3

EQ Coder for Normal Meshes

3.1 Mixture models for the wavelet coefficients

The normal mesh data consists of the mesh connectivity and the mesh geometry. A wavelet transform of the normal mesh data results in the scaling coefficients corresponding to the base vertices and the wavelet coefficients corresponding to the finer level vertices. Each scaling coefficient is a vector of the coordinates of the base vertices and each wavelet coefficient is a vector of one normal and two tangential components. We propose a context based coder for the wavelet coefficients where the wavelet coefficients are modeled using the Generalized Gaussian distribution (GGD) model [22].

The probability density function of a GGD random variable $X$ is given below

$$f_{(\nu, \mu, \sigma)}(X = x) = \frac{\nu \eta(\nu, \sigma)}{2\Gamma(1/\nu)} e^{-(\eta(\nu, \sigma)|x-\mu|)^\nu}$$

(3.1)

where $\eta(\nu, \sigma) = \frac{1}{\sigma} \sqrt{\frac{\Gamma(3/\nu)}{\Gamma(1/\nu)}}$ and $0 < \nu \leq 2$. The GGD density function depends on three parameters, the shape $\nu$, the mean $\mu$ and the standard deviation $\sigma$. The Laplacian and the Gaussian distributions are special cases of the GGD corresponding to $\nu = 1$ and $\nu = 2$ respectively.

Each component of the wavelet coefficient is modeled using an independent zero-mean GGD random variable. The shape $\nu$ is assumed to be fixed at each level for each component of the wavelet coefficient and the standard deviation $\sigma$ is assumed to
be slowly spatially-varying. In the Estimation-Quantization (EQ) algorithm, we first estimate the parameters of the model and then quantize the coefficients based on the estimated GGD model. The EQ algorithm can therefore be split into two steps, the estimate step and the quantize step. The shape $\nu$ is estimated from the data at each level for each component and the standard deviation $\sigma$ is estimated from the data using the local context information that is known to both the encoder and the decoder. The estimate step is used to estimate these parameters and is discussed in Section 3.2. Once the parameters are estimated, the wavelet coefficients are quantized using Rate-Distortion (R-D) optimization in the quantize step discussed in Section 3.3. The implementation details of the EQ algorithm are discussed in the next Chapter.

3.2 Estimate step

3.2.1 Estimating the shape parameter

The shape parameter is assumed to be fixed for each level and for each of the three components of the wavelet coefficients. The wavelet coefficients normalized by the corresponding standard deviation values of each coefficient form a unit-variance GGD with a fixed shape parameter for each component at each level. Therefore, the value of the shape parameter is estimated using the normalized values of the coefficients at each level for each component. The values of the shape parameters are then sent to the decoder, since the decoder does not have access to the original values of the wavelet coefficients.
3.2.2 Estimating the standard deviation parameter

The standard deviation of each wavelet coefficient component is assumed to depend on the local neighborhood of the corresponding vertex. The quantized values in the causal neighborhood are used to estimate the standard deviation for each wavelet coefficient component, since the decoder has access to only the quantized values of the wavelet coefficients in the causal neighborhood.

Consider a vertex $i$ with the causal neighborhood $\mathcal{N}_i$. Let $x_i$ be one of the components of the wavelet coefficient for the vertex $i$, $x_j$ be the corresponding component of the wavelet coefficient for vertex $j$, where $j \in \mathcal{N}_i$, $\hat{x}_j$ be the quantized value of $x_j$ and $l_j$ and $r_j$ be the left and right boundaries of the quantizer bin used to quantize $x_j$. The maximum likelihood (ML) estimate of the standard deviation $\hat{\sigma}_i$ for $x_i$, assuming that the coefficients $x_j$ in the local neighborhood are drawn independently from the GGD with standard deviation $\hat{\sigma}_i$, is obtained from the quantized values in the causal neighborhood $\mathcal{N}_i$ as shown below

$$
\hat{\sigma}_i = \arg \max_{\sigma} \prod_{j \in \mathcal{N}_i} P(l_j \leq x \leq r_j) \\
= \arg \max_{\sigma} \sum_{j \in \mathcal{N}_i} \log \int_{l_j}^{r_j} \left[ \frac{\nu \eta(\nu, \sigma)}{2 \Gamma(1/\nu)} \right] \exp(-[\eta(\nu, \sigma)|x|^\nu) \, dx 
$$

(3.2)

where $\nu$ is the shape parameter, $\eta(\nu, \sigma) = \frac{1}{\sigma} \left[ \frac{\Gamma(3/\nu)}{\Gamma(1/\nu)} \right]^{1/2}$ and $P(l \leq x \leq r)$ is the probability that the GGD random variable lies in the interval $(l, r)$.

The expression in equation (3.2) is differentiated with respect to $\sigma$ and set to zero to obtain the value of $\hat{\sigma}_i$. The resulting equation is rearranged to get the following
non-linear equation

\[ \hat{\sigma}_i = \left[ \frac{1}{N} \sum_{j=1}^{N} \nu |\eta(\nu, 1)|^\nu \int_{r_j}^{l_j} x |x|^\nu \exp(-|\eta(\nu, \hat{\sigma}_n)|x|^\nu) \, dx \right]^{1/\nu} \]  

(3.3)

where \( N \) is the number of causal neighbors for vertex \( i \). The equation can be solved iteratively starting with an initial guess obtained by the following approximation.

\[ \frac{\int_{r_j}^{l_j} x |x|^\nu \exp(-|\eta(\nu, \hat{\sigma}_n)|x|^\nu) \, dx}{\int_{r_j}^{l_j} \exp(-|\eta(\nu, \hat{\sigma}_n)|x|^\nu) \, dx} \approx |\hat{x}_j|^\nu. \]  

(3.4)

This approximation holds when \( \hat{x}_j \neq 0 \), for which \( P(l_j \leq x_j \leq r_j) \) is small. Therefore, the approximation in equation (3.4) is used for all the coefficients in the causal neighborhood to obtain an initial estimate \( \hat{\sigma}_{i0} \) for \( \sigma \). In the next iteration, the approximation is used only for those coefficients in the causal neighborhood that have not been quantized to zero and the initial estimate \( \hat{\sigma}_{i0} \), is used for the coefficients that have been quantized to zero.

The Gaussian case (\( \nu = 2 \)) is used to simplify the computations of the ML estimate of \( \hat{\sigma}_i \). In the Gaussian case, equation (3.3) gets simplified to the following equation

\[ \hat{\sigma} = \left[ \frac{1}{N} \sum_{j=1}^{N} \int_{r_j}^{l_j} x^2 \exp(-x^2/2\hat{\sigma}_n^2) \, dx \right]^{1/2} \]  

(3.5)

and the approximation in equation (3.4) gets reduced to the following approximation

\[ \frac{\int_{r_j}^{l_j} x^2 \exp(-x^2/2\hat{\sigma}_n^2) \, dx}{\int_{r_j}^{l_j} \exp(-x^2/2\hat{\sigma}_n^2) \, dx} \approx \hat{x}_j^2. \]  

(3.6)

### 3.3 Quantize step

Once we obtain an estimate for the standard deviation, the coefficient is quantized using a scalar deadzone quantizer. The bin-widths \( \delta_0 \) and \( \delta_1 \) characterize the deadzone
quantizer, where \( \delta_0 \) is the width of the zero bin and \( \delta_1 \) is the width of the nonzero bins. The choice of the quantizer is made based on R-D optimization.

### 3.3.1 Rate-Distortion optimization

The R-D optimization is based on Lagrangian optimization with the Lagrangian cost function for each component of the wavelet coefficient of vertex \( i \) given by

\[
J_i = r_i + \lambda d_i,
\]

where \( \lambda \) is the Lagrangian parameter or the R-D slope, and \( r_i \) and \( d_i \) are the corresponding rate and distortion terms respectively. The rate \( r_i \) is expressed in terms of the entropy associated with the corresponding bin probabilities of the quantizer and the distortion \( d_i \) is expressed in terms of squared quantization error associated with each quantizer.

Consider a coefficient \( y \) quantized to \( \hat{y}_j \) corresponding to bin \( j \) of the quantizer \((\delta_0, \delta_1)\). The rate \( r \) and distortion \( d \) for this coefficient are expressed as

\[
\begin{align*}
    r(y, \delta_0, \delta_1) &= -\log_2(p_{\nu, \mu, \sigma}(j, \delta_0, \delta_1)) \\
    d(y, \delta_0, \delta_1) &= (y - \hat{y}_j)^2
\end{align*}
\]

where \( p_{\nu, \mu, \sigma}(j, \delta_0, \delta_1) \) is the bin probability and \( \hat{y}_j \) is the quantization value associated with the bin \( j \) and the quantizer \((\delta_0, \delta_1)\). Note that the bin probability and the quantization value depend on the probability density function and hence on the shape
and the standard deviation values for the current coefficient as shown below

$$p(\nu, \mu, \sigma)(j, \delta_0, \delta_1) = \int_{r_j}^{l_j} f(\nu, \mu, \sigma)(X = x) \, dx$$

$$\tilde{y}_j = \frac{\int_{r_j}^{l_j} x \, f(\nu, \mu, \sigma)(X = x) \, dx}{\int_{r_j}^{l_j} f(\nu, \mu, \sigma)(X = x) \, dx}$$

where $l_j$ and $r_j$ are the left and right bin boundaries respectively for the bin $j$ and the quantizer $(\delta_0, \delta_1)$ and $f(\nu, \mu, \sigma)(X)$ is the GGD probability density function with shape $\nu$, mean $\mu$ and standard deviation $\sigma$.

### 3.3.2 Rate-Distortion curves

Based on the above definitions of rate and distortion, the choice of the quantizer for a coefficient $y$ is made using the R-D curve corresponding to the shape and the standard deviation values associated with the coefficient $y$. The R-D curves are constructed for the unit variance case for a set of different shape parameters using the definitions of rate and distortion given by equation (3.12) and equation (3.13) respectively.

$$r(\delta_0, \delta_1) = \sum_{j=-\infty}^{0} \int_{-\delta_0/2+(j-1)\delta_1}^{-\delta_0/2+j\delta_1} f(\nu,0,1)(y) \, dy + \int_{-\delta_0/2}^{\delta_0/2} f(\nu,0,1)(y) \, dy$$

$$d(\delta_0, \delta_1) = \sum_{j=0}^{\infty} \int_{\delta_0/2+j\delta_1}^{\delta_0/2+(j+1)\delta_1} (y - \tilde{y}_j)^2 f(\nu,0,1)(y) \, dy + \int_{-\delta_0/2}^{\delta_0/2} y^2 f(\nu,0,1)(y) \, dy$$

where $f(\nu, \mu, \sigma)$ is the GGD density function with shape $\nu$, mean $\mu$ and standard deviation $\sigma$ and $\tilde{y}_j$ is the quantized value for the bin $j$. The quantized value $\tilde{y}_j$ for bin $j$
is computed using the GGD density function for each quantizer as shown below

\[ \tilde{y}_j = \frac{\int_{r_j}^{l_j} y f_{(\nu,0,1)}(y) \, dy}{\int_{r_j}^{l_j} f_{(\nu,0,1)}(y) \, dy} \]

where \(l_j\) and \(r_j\) are given by

\[
(l_j, r_j) = \begin{cases} 
-\delta_0/2 + (j - 1)\delta_1, -\delta_0/2 + j\delta_1 & \text{if } j < 0; \\
-\delta_0/2, \delta_0/2 & \text{if } j = 0; \\
\delta_0/2 + j\delta_1, \delta_0/2 + (j + 1)\delta_1 & \text{if } j > 0.
\end{cases}
\]

The distortion values of the R-D curves for the unit-variance case are scaled by \(\sigma^2\) to obtain the R-D curves for density functions with variance \(\sigma^2\) and the quantizers for the unit-variance case are scaled by \(\sigma\) to obtain the quantizers for the density function with variance \(\sigma^2\). Thus, we need to compute R-D curves only for the unit-variance case. The R-D curves are stored in the form of R-D tables indexed by slope \(\lambda\).

For a given value of a R-D slope \(\lambda\) and a coefficient \(y\) with a variance \(\sigma^2\), the corresponding slope on the R-D curve for the unit variance GGD is \(\lambda/\sigma^2\). This slope is associated with a quantizer \((\delta_0, \delta_1)\) in the R-D table. The quantizer \((\delta_0, \delta_1)\) corresponding to the slope \(\lambda/\sigma^2\) is scaled by \(\sigma\) to obtain the quantizer \((\sigma\delta_0, \sigma\delta_1)\) for the current coefficient \(y\) with a variance \(\sigma^2\).

### 3.3.3 Entropy coding

Once the quantizer \((\delta_0, \delta_1)\) and the GGD parameters \((\nu, \sigma)\) for a given coefficient are known, the corresponding bin probabilities can be easily computed and are used to encode the quantization symbol with an entropy coder such as the arithmetic coder.
[23]. The bin probabilities for each quantizer from the R-D table are computed offline to speed up the algorithm.

3.4 Unpredictable set

A problem with the EQ algorithm is that if all the causal neighbors of the current vertex have been quantized to zero, then the variance estimate \( \hat{\sigma} \) for the current coefficient is zero. Therefore, the current coefficient is automatically quantized to zero. This phenomenon could propagate through the rest of the coefficients at the same level leading to instability in the EQ algorithm.

3.4.1 Characterizing the unpredictable set

A simple solution to this problem is to place the coefficients with all causal neighbors quantized to zero into a separate set called the unpredictable set. The coefficients in the unpredictable set are modeled using a zero-mean GGD with a fixed shape and a fixed variance. These parameters are estimated from the wavelet coefficients in the unpredictable set and are sent to the decoder. The coefficients that do not belong to the unpredictable set are classified as the predictable set. The initial estimates for the shape parameters of the predictable and the unpredictable sets and the initial estimate for the standard deviation parameter of the unpredictable set are obtained based on the initial partitioning of the coefficients into the predictable and the unpredictable sets at each level.
3.4.2 Initial partitioning of the coefficients

The coefficients are initially partitioned into the predictable and the unpredictable sets using a thresholding technique. Note that for a Gaussian random variable, the ML estimate $\hat{\sigma}$ from a single observation $x$ is equal to the absolute value of the sample $x$.

$$\hat{\sigma} = |x|$$  \hspace{1cm} (3.14)

Quantizing the coefficient $x$ with the quantizer corresponding to $\hat{\sigma} = |x|$ and a R-D slope $\lambda$ is optimal for the Gaussian model. This corresponds to quantizing $x/\hat{\sigma} = \pm 1$ with a R-D slope $\lambda' = \lambda/\hat{\sigma}^2$. The unit variance Gaussian case has a R-D slope $\lambda_t = 0.264$ that corresponds to a quantizer that just quantizes $\pm 1$ to zero and has a zero-bin width of 2.0. A quantizer with a higher or equal slope

$$\frac{\lambda}{\hat{\sigma}^2} \geq \lambda_t$$  \hspace{1cm} (3.15)

quantizes $\pm 1$ to zero and a quantizer with a lower slope

$$\frac{\lambda}{\hat{\sigma}^2} < \lambda_t$$  \hspace{1cm} (3.16)

quantizes $\pm 1$ to a nonzero value. Therefore, the threshold for the coefficient $x$ is obtained by combining the equation (3.14), equation (3.15) and equation (3.16) and is given by

$$t = \sqrt{\frac{\lambda}{\lambda_t}}.$$

Thus, a coefficient $x \leq t$ is quantized to zero. Based on this thresholding technique, the initial partition of the coefficients into predictable and unpredictable sets is obtained.
3.4.3 Tangential wavelet coefficients

The wavelet coefficients for each vertex consists of one normal component and two tangential components defined in the local coordinate system. For a smooth surface, a small displacement in a direction normal to the surface changes the original surface much more compared to the same displacement in a direction tangential to the surface. Therefore, the tangential components of the wavelet coefficients are encoded using higher R-D slope $\lambda$ compared to the normal components.

Another difference between the tangential components and the normal components of wavelet coefficients from a normal mesh is that the tangential components are mostly zero. Thus there is very little context information for the tangential component of a given vertex. Therefore, all the tangential components are classified as an unpredictable set and are characterized by a single shape and standard deviation parameter at each level.

3.5 Vertex neighborhood

The local neighborhood of the current vertex is used to estimate the standard deviation for the current coefficient. If the local neighborhood is small, then it does not provide enough information to estimate the standard deviation. On the other hand if the neighborhood is large, then the spatially dependence is lost. Figure 3.1 shows a simple pixel neighborhood used in the 2D image case. The mesh data has no similar obvious choice for a vertex neighborhood. Figure 3.2 shows a simple four-neighbor rule but it is too small to obtain a reasonable estimate for the standard deviation. Therefore we increase the neighborhood to include the next two rings or layers of neighboring vertices as shown in figure 3.3 and figure 3.4. Figure 3.3 shows the neigh-
borhood for a vertex that lies on an edge connected by regular vertices and figure 3.4 shows the neighborhood for a vertex that lies on an edge connected by extraordinary vertices.

![Image Grid](image_grid.png) ![Diagram](diagram.png)

**Figure 3.1** Simple $3 \times 3$ pixel neighborhood in an image.  
**Figure 3.2** Simple four-neighbor rule for a regular vertex in a mesh.

### 3.6 Vertex ordering

The original ordering of the vertices in the mesh is arbitrary. Therefore, a global ordering of the vertices is not obvious as is the case for 2D images where a simple raster-scanning is used for the pixel values as shown in figure 3.5. A more structured ordering for the vertices is necessary in order to improve the number of causal neighbors in the local vertex neighborhood and to preserve the semi-regular connectivity of the normal mesh. We propose an ordering for the vertices starting with the vertices at the coarse level and then proceeding to finer levels, where at each level, each base triangle is scanned separately. Therefore, the global ordering of the vertices is reduced to ordering of the base triangles and ordering of the vertices inside each base
triangle.

3.6.1 Vertex ordering inside a base triangle

Figure 3.6 shows the ordering of vertices inside each base triangle, where the ordering starts from bottom-left and moves towards right and then up. The vertices from the neighboring base triangles are also considered while constructing the vertex neighborhoods for the vertices of the current base triangle. A status variable for each vertex keeps track of whether the current vertex is a scanned vertex or an unscanned vertex. Based on this status variable, the causal neighborhood is derived from the local vertex neighborhood. This vertex ordering inside the base triangles preserves the semi-regular connectivity of the normal mesh.
3.6.2 Base triangle ordering

Figure 3.7 shows the ordering of the base triangles. The first base triangle $T_i$ is randomly selected. Once the current base triangle $T_i$ is scanned, the next base triangle $T_{i+1}$ is selected from the list of the unscanned neighboring base triangles of the current base triangle $T_i$. If there are no unscanned neighboring base triangles, then we return to the first base triangle and repeat the process until all the base triangles are included in the list.

3.6.3 Base triangle orientation

The orientation of the base triangle is selected in such a way that it maximizes the number of vertices in the causal neighbors. Figure 3.8 illustrates the causal neighborhoods for different vertices of the base triangle at a particular level using the simple four-neighbor rule shown in figure 3.2. $a$, $b$ and $d$ are vertices on horizontal, diagonal and vertical edges respectively and $c$ is an interior vertex. Vertex $a$ has no causal
Figure 3.6 Vertex ordering inside base triangles.
neighbors, \( d \) has one causal neighbor and \( b \) and \( c \) have two causal neighbors each, assuming that the all the neighboring base triangles \( N1, N2 \) and \( N3 \) are unscanned.

The vertices along the horizontal edge have the smallest size of causal neighborhoods followed by the vertices on the vertical edge and then the diagonal edge vertices and the interior vertices, assuming that all the neighboring base triangles are unscanned. Therefore, the orientation of the base triangle is changed in such a way that scanned adjacent base triangles, if any, are along the horizontal, vertical and diagonal edges in that order of priority.

### 3.7 Scaling coefficients

The coordinates of the vertices of the base mesh form the scaling coefficients. The scaling coefficients are uniformly quantized using a variable bit-rate. The choice of bits-per-vertex for the base mesh vertices is done in such a way that it is much higher
Figure 3.8  Causal neighborhood using a simple four-neighbor rule.

compared to bits-per-vertex for the finer level vertices, because the scaling coefficients have a significantly higher global error contribution compared to the wavelet coefficients.

3.8  Mesh connectivity

The vertex ordering is done in such a way that the semi-regular connectivity of the normal mesh is preserved. The base triangle ordering and orientation are derived from the base mesh connectivity. The vertex ordering inside each base triangle is fixed once the ordering and orientation of base triangle is known. Therefore, only the base mesh connectivity is required to be transmitted to the decoder.
The base mesh connectivity is encoded using a TG coder [20]. The TG coder is a common encoder that is used for compressing triangular meshes giving near optimal performance in compressing connectivity information for irregular meshes and optimal performance for regular meshes. The TG coder also compresses the vertex information, however in this thesis it is used to encode the base mesh connectivity only. The TG coder encodes the connectivity information as a traversal of the vertices and it maintains an active list of vertices and edges that have already been encoded. A deterministic scheme is used to add edges and vertices with respect to a pivot vertex. The pivot vertex is moved along the active list as the active list increases. A valence symbol is output for each vertex and the error in valence is encoded using Huffman coding [24]. Therefore, for a regular mesh, the bit-rate for connectivity information vanishes to zero.
Chapter 4

Algorithm Implementation Details

4.1 Mesh file format

Normal meshes are stored in the DAT file format [6]. A wavelet transform of the normal mesh is also stored in the DAT file format. The wavelet transform in the DAT file format has two parts, the mesh geometry and the mesh connectivity. The geometry part lists the vertices by level starting with the base level. The base mesh vertices or the scaling coefficients are expressed using the absolute coordinates and the vertices at the finer levels are expressed in terms of the wavelet coefficients using the local coordinates. Each wavelet coefficient consists of one normal component and two tangential components. The different parts of the normal mesh data that need to be encoded are the base mesh connectivity, the scaling coefficients and the wavelet coefficients.

4.2 Encoding mesh connectivity

4.2.1 TG coder correction

The original order of the base triangles needs to be preserved as it is used at the decoder to determine the ordering and the orientation of the base triangles. The TG coder does not preserve the original order of the base triangles. However, if the TG coder is used to re-encode the original output of the TG coder, then the order of the
base triangles in the original output is preserved in the final output of the TG coder. Therefore, before using the EQ coder, the triangles in the original mesh are reordered in such a way that encoding the base mesh connectivity of the reordered mesh with the TG coder preserves the ordering of the base triangles in the mesh input to the EQ coder.

4.2.2 Array representation for mesh connectivity

The mesh connectivity for each base triangle is represented separately in a triangular array form. Figure 4.1 (a) shows a base triangle \{1, 2, 3\} that has been subdivided twice. Figure 4.1 (b) shows the array representation of the base triangle in (a) and (c) and (d) show the same base triangle after one level of subdivision and before any subdivision respectively. The array representation assigns some \textit{structure} to the original list of arbitrarily ordered triangles.

4.2.3 Base mesh connectivity

The base mesh connectivity is encoded using the TG coder. Since the original mesh connectivity is reordered, the TG coder preserves the ordering of the base triangles. At the decoder, a TG decoder is used to decode the base mesh connectivity. The decoder uses the base mesh connectivity to determine the ordering and orientation of the base triangles that was used at the encoder.
Figure 4.1 Array representation of a base triangle.
4.2.4 Finer level mesh connectivity

The vertices at each level are scanned separately starting with the base level and proceeding to the finer levels. At each level, each base triangle is scanned separately. The vertex ordering preserves the semi-regular connectivity of the normal mesh. Therefore, it is not necessary to send the finer level mesh connectivity to the decoder.

4.2.5 Vertex neighborhoods

The connectivity information of the normal mesh is represented using the array representation as discussed in section 4.2.2. The neighborhood definition discussed in section 3.5 is used to determine the vertex neighborhoods from the mesh connectivity. The vertices that are directly connected to the current vertex form the first layer of neighbors for the current vertex. The vertices that are directly connected to the neighbors of the current vertex are then added to the neighborhood of the current vertex and then the remaining neighbors shown in the figure 3.3 and figure 3.4 are added to the neighborhood of the current vertex.

4.3 Encoding mesh geometry

4.3.1 Scaling and wavelet coefficients

The scaling coefficients are quantized uniformly. The wavelet coefficients are encoded using the EQ algorithm. A R-D slope parameter $\lambda$ is selected for the entire mesh. The coefficients at each level are partitioned into predictable and unpredictable sets. The unpredictable sets are characterized by a zero-mean GGD with a fixed shape
and a fixed standard deviation. The estimate and quantize steps are used to estimate the standard deviation for the coefficients in the predictable set. The coefficients normalized with the standard deviation form a unit-variance GGD with the shape parameter of the predictable set. Therefore, the shape parameter for the predictable set is obtained from this unit-variance GGD. The estimation of the shape $\nu_U$ and the standard deviation $\sigma_U$ for the unpredictable set and the shape $\nu_P$ for the predictable set is discussed in section 4.3.4.

A threshold technique is applied to the wavelet coefficients in the first iteration to partition the coefficients into predictable and unpredictable sets. Based on this partition, the initial estimates for the shape and the standard deviation of the unpredictable set and the shape for the predictable set are obtained at each level. These initial estimates are then used in next iteration of the estimate and quantize step of the EQ algorithm. The EQ algorithm dynamically partitions the coefficients into predictable and unpredictable sets and these partitions are used for the following iterations of the estimate and the quantize steps.

### 4.3.2 Estimate step

In the current iteration, the shape parameters from the previous iteration are used to estimate the standard deviation $\hat{\sigma}$ for each coefficient using equation (3.5) and equation (3.6). If $\hat{\sigma}$ for the current coefficient is zero, then the coefficient is classified as unpredictable and the standard deviation $\sigma_U$ and the shape $\nu_U$ estimates of the unpredictable set from the previous iteration are used in the quantize step. If $\hat{\sigma}$ is nonzero, then the coefficient is classified as predictable and the current standard deviation estimate $\hat{\sigma}$ and shape estimate $\nu_P$ from the previous iteration are used in
the quantize step. The new partitioning of the coefficients into predictable and unpredictable sets is then used at the end of the current iteration to estimate the new values for the parameters \( \nu_U, \sigma_U \) and \( \nu_P \).

4.3.3 Quantize step

The value of the shape from the estimate step is used to identify the corresponding unit-variance GGD R-D curve for the current coefficient. The value of the standard deviation \( \sigma \) from the estimate step is used to find the slope \( \lambda/\sigma^2 \) of the R-D curve. The R-D tables are used to find the quantizer \((\delta_0, \delta_1)\) that corresponds to this slope. The quantizer bins are scaled by \( \sigma \) to obtain the quantizer \((\sigma\delta_0, \sigma\delta_1)\) for the current coefficient. This quantizer is used to obtain the quantization index for the current coefficient. The arithmetic coder uses the bin probabilities of the corresponding quantizer to encode the quantization index of the current coefficient.

4.3.4 Estimating the GGD parameters

The technique proposed in [25] is used to estimate the shape and standard deviation parameters for a GGD data set. Let \( X \) be a random variable for a zero-mean GGD data set. The expectation of the absolute value of the random variable \( X \) is given by

\[
E[|X|] = \sigma \frac{\Gamma(2/\nu)}{\sqrt{\Gamma(3/\nu)\Gamma(1/\nu)}}
\]

and the variance of the random variable is given by

\[
s^2 = E[X^2].
\]
The ratio of $E[X^2]$ and $E[|X|]$ can be simplified to get the following expression

$$
\frac{E[X^2]}{E[|X|]} = \frac{\Gamma(1/\nu)\Gamma(3/\nu)}{\Gamma^2(2/\nu)}.
$$

(4.1)

The ratio in equation (4.1) is a function of the shape parameter alone and can be denoted as $\rho(\nu)$. This ratio forms a one-to-one mapping between $\rho$ and $\nu$ and therefore the function $\rho(\nu)$ is invertible. The value of $\rho$ is obtained from the given data set and the inverse function $\rho^{-1}$ is used to find the shape $\nu$.

### 4.3.5 Tangential wavelet coefficients

The tangential components are classified as unpredictable sets. The shape and the standard deviation values for these unpredictable sets, $\nu_U$ and $\sigma_U$, are used to encode the tangential components. As the contribution of the tangential components to the global error is smaller compared to the normal components, a higher value of the R-D slope $\lambda$ is used for the tangential components.

### 4.3.6 Model fit for data

Figures 4.2, 4.3 and 4.4 show the GGD fit obtained from the EQ algorithm for the wavelet coefficients of the Venus head normal mesh. The normal components of the wavelet coefficients at each level are partitioned into predictable and unpredictable sets. The unpredictable sets are characterized using a zero-mean GGD with a fixed shape and standard deviation. Figure 4.2 shows the GGD fit for one of the tangential components of the wavelet coefficients. Figure 4.3 shows the GGD fit for the unpredictable normal coefficients. These figures show the coefficients in the unpredictable sets normalized by the value of the standard deviation $\sigma_U$ used to characterize the
corresponding unpredictable set.

![GGD fit for the unpredictable tangential coefficients.](image)

**Figure 4.2** GGD fit for the *unpredictable* tangential coefficients.

The predictable normal coefficients normalized by the corresponding standard deviation estimate obtained from the EQ algorithm form a unit-variance GGD. Figure 4.4 shows the GGD fit for the unit-variance GGD formed by the predictable normal coefficients.
4.4 Decoding mesh connectivity and mesh geometry

4.4.1 Decoding mesh connectivity

A TG decoder is used to decode the base mesh connectivity at the decoder. The reconstructed base mesh connectivity is used to find the ordering and the orientation of the base triangles that was used at the encoder. As the vertex ordering that is used inside each base triangle preserves the semi-regular connectivity of the normal mesh, the connectivity at finer levels can be easily reconstructed. The neighborhoods of all the vertices can be easily computed from the mesh connectivity.
Figure 4.4  GGD fit for the *predictable* normal coefficients.
4.4.2 Decoding mesh geometry

The base vertices are obtained using inverse uniform quantization. The shape $\nu_U$ and standard deviation $\sigma_U$ values for the unpredictable sets and the shape $\nu_P$ values for the predictable sets are sent to the decoder as side information. These values are then used to identify the shape, the standard deviation and the corresponding quantizer for all the coefficients using the estimate and quantize steps of the EQ algorithm. The quantization index obtained from the arithmetic decoder is used to predict the wavelet coefficients.

The decoded mesh connectivity, base vertices and wavelet coefficients are arranged in the DAT file format to get the reconstructed wavelet transform of the normal mesh.
Chapter 5

Results and Analysis

In this Chapter, we first describe the error metrics and the software tools used to make the error measurements. The details of the data sets used in the error measurements are given. We analyze the performance of the EQ coder and compare it with the zerotree (ZT) coder for normal meshes.

5.1 Error metrics

Let $M_1$ and $M_2$ be the two meshes being compared and let $S_1$ and $S_2$ be the corresponding surfaces described by these meshes. Note that the number of vertices and triangles in the two meshes that are being compared need not be the same. Therefore a one-to-one mapping between the vertices of the first mesh and the vertices of the second mesh or a one-to-one mapping between the triangles of the first mesh and the triangles of the second mesh cannot be guaranteed. This means that standard error metrics such as the mean squared error metric that is used for images cannot be used for the mesh data. In the following section, we discuss a volume based error metric defined for meshes that uses the approximate distance. This error metric is equivalent to mean squared and mean distances between the two surfaces.
Approximate distance

The error metrics used in our work are based on the approximate distance. The approximate distance from the surface $S_2$ to a point $p$ on surface $S_1$ is given by

$$e(p, S_2) = \min_{p' \in S_2} d(p, p')$$

where $e(p, S_2)$ is the approximate distance and $d(p, p')$ is the Euclidean distance between the two points $p$ and $p'$. The approximate distance is used to define the Hausdorff distance and the mean and the mean squared distances between the two surfaces.

Hausdorff distance

The distance between the two surfaces $S_1, S_2$ is defined as

$$E(S_1, S_2) = \max_{p \in S_1} e(p, S_2).$$

This distance measure is also known as the directed Hausdorff distance and is asymmetric in nature (Appendix A). The maximum of $E(S_1, S_2)$ and $E(S_2, S_1)$ can be used to define a symmetric Hausdorff distance.

Mean and mean squared distance

The mean distance between the two surfaces $S_1, S_2$ is defined as the surface integral of the approximate distance $e(p, S_2)$ divided by the area of surface $S_1$ and is given by

$$E_m(S_1, S_2) = \frac{1}{|S_1|} \int_{S_1} e(p, S_2) \, ds \quad (5.1)$$
where $|S_1|$ is the area of surface $S_1$. The root mean squared distance between the two surfaces $S_1$ and $S_2$ is defined as

$$E_{rms}(S_1, S_2) = \sqrt{\frac{1}{|S_1|} \int_{S_1} e^2(p, S_2) \, ds} \quad (5.2)$$

where $E_{rms}(S_1, S_2)$ is the root mean squared distance or the RMS distance.

In this thesis, we use the mean and the RMS distances to measure the error between two meshes. The integrals in equation (5.1) and equation (5.2) are computed by sampling the faces of the first mesh $M_1$ and computing the approximate distances from surface $S_2$ to the sampling point on surface $S_1$. Metro [26, 27], MeshDev [28] and Mesh [29] are some of the software tools that are used to measure the error between two meshes. The Metro and the MeshDev tools are briefly explained in the next section.

### 5.2 Error tools

**Metro error**

The Metro tool [26, 27] compares two triangular meshes which describe the same surface and gives the volume difference between them. The tool accepts input meshes in the Open Inventor format [4] and error can be measured in terms of the symmetric mean distance, the symmetric RMS distance and the volume difference for closed surfaces. Appendix B.1 shows a sample Metro output comparing two Venus head meshes.
MeshDev error

The MeshDev tool [28] compares two triangular meshes in terms of the geometry deviation or the attribute deviation for attributes such as texture, color. In this thesis, we use the MeshDev tool to measure the geometry deviation between the two meshes. The geometry deviation corresponds to the mean distance between the two surfaces. The tool accepts input meshes in Michael Garland’s Simple Model File (SMF) format [5]. Appendix B.2 shows a sample MeshDev output comparing two horse meshes.

5.3 Data sets used

We use the Venus head, the horse and the rabbit normal meshes for analyzing the performance of the EQ coder. Figure 5.1 shows these models and table 5.1 gives the details of the normal meshes and the original irregular meshes for these models. The original vertices and triangles in table 5.1 are the number of vertices and triangles in the original irregular mesh.

![Venus head, horse, and rabbit models](image)

**Figure 5.1** The Venus head, the horse and the rabbit models.
Table 5.1 The Venus head, the horse and the rabbit normal meshes.

<table>
<thead>
<tr>
<th>Number</th>
<th>Venus mesh</th>
<th>Horse mesh</th>
<th>Rabbit mesh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Levels</td>
<td>6</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Base vertices</td>
<td>42</td>
<td>112</td>
<td>71</td>
</tr>
<tr>
<td>Vertices</td>
<td>163842</td>
<td>112642</td>
<td>70658</td>
</tr>
<tr>
<td>Triangles</td>
<td>327680</td>
<td>225280</td>
<td>141312</td>
</tr>
<tr>
<td>Original vertices</td>
<td>50002</td>
<td>48485</td>
<td>67038</td>
</tr>
<tr>
<td>Original triangles</td>
<td>100000</td>
<td>96966</td>
<td>134074</td>
</tr>
</tbody>
</table>

5.4 Error measurements

The reconstructed normal mesh is compared with the original irregular mesh to obtain the error measurements. Normal remeshing introduces remeshing error and the EQ coder introduces compression error. Therefore, the error between the reconstructed normal mesh and the original irregular mesh is a combination of the effects of normal remeshing and EQ compression.

The original irregular mesh and the reconstructed normal mesh are compared using the Metro and MeshDev tools. The Metro tool is used to measure the RMS distance in terms of the absolute value and the error values expressed as a percentage of the bounding box diagonal. The bounding box diagonal is the longest diagonal of the box that bounds the original surface. The MeshDev tool is used to measure the mean geometry deviation. The Metro RMS distance values are given in terms of $10^{-4}$ and the MeshDev mean geometry mean deviation values are given in terms of $10^{-3}$. The PSNR values are obtained as a ratio of the bounding box diagonal and the RMS distance measurement from the Metro tool. The error values and the PSNR values
are plotted against bits-per-vertex (bpv). The bits-per-vertex values are expressed in terms of the total number of bits used per vertex of the original irregular mesh.

5.5 Results

5.5.1 Performance of the EQ coder

The Venus head, the horse and the rabbit normal meshes are encoded using the EQ coder. Figures 5.2, 5.3, 5.4 show the PSNR values, the mean and the RMS errors measured for these meshes along with the remeshing error. From the figures, we can see that the error values tend towards the remeshing error for higher bits-per-vertex values.

We give some bounds for the performance of the EQ coder. We obtain the lower bound by encoding the coefficients using the same quantizer for all the coefficients at each level for the normal and tangential components without using the local context information. The coefficients are modeled as a GGD random variable with the shape and variance parameters fixed for each level. The shape and variance parameter estimates are obtained for each level and for each of the three components of the wavelet coefficients. The reconstructed meshes are compared with the original irregular mesh.

The pre-quantized coefficient values from the complete neighborhood of each vertex are used to estimate the variance for the current coefficient to get the upper bounds for the EQ performance. This algorithm is not practical, because the decoder does not have the complete neighborhood information, nor does it have the pre-quantized values of the coefficients. However, we use this algorithm to give an upper bound
Figure 5.2  Metro error using the EQ coder for the Venus head, the horse and the rabbit models.
Figure 5.3 Metro PSNR in $dB$ using the EQ coder for the Venus head, the horse and the rabbit models.
Figure 5.4  MeshDev error using the EQ coder for the Venus head, the horse and the rabbit models.
Figure 5.5  GGD fit for the normal coefficients of the Venus head normal mesh normalized by the standard deviation of the complete neighborhood.
for the EQ performance, based on the model assumed for the wavelet coefficients in Section 3.1. The GGD data fit obtained for this model is shown in figure 5.5, and can be compared with EQ model data fit shown in figures 4.3, 4.4 and 4.2. Under this model, most of the coefficients have non-zero neighborhoods. The mesh is encoded with this algorithm and then the quantized values are used to reconstruct the mesh. The reconstructed mesh is compared with the original irregular mesh.

Figures 5.6 (a) - (d) show the upper and the lower bounds for the Venus head normal mesh. We can see from the figures that the EQ performance is very close to the upper bound. All the above figures are plots based on the unfolded butterfly wavelet transform.

5.5.2 Comparison of the EQ and the zerotree coders

Now, we compare the performance of the EQ coder with the zerotree coder. The normal meshes are first compressed with the EQ and zerotree coders at different bits-per-vertex, and then reconstructed. The reconstructed meshes are compared with the original irregular. Figures 5.7, 5.8, 5.9 show the plots.

The PSNR values are shown in tables 5.2, 5.3 and 5.4. The unfolded version of the butterfly wavelet transform is used for both the EQ and zerotree coders.

5.5.3 Comparison of different wavelet transforms

The Venus head, the horse and the rabbit normal meshes are constructed using the butterfly subdivision scheme and therefore if we use the same wavelet transform, i.e.,
Figure 5.6  Bounds for the EQ coder (the Venus head normal mesh) in terms of the Metro error, the Metro PSNR and the MeshDev error.
Figure 5.7  Comparing the EQ and zerotree coders using Metro error, Metro PSNR and MeshDev error for the Venus head normal mesh.
Figure 5.8 Comparing the EQ and zerotree coders using Metro error, Metro PSNR and MeshDev error for the horse normal mesh.
Figure 5.9 Comparing the EQ and zerotree coders using Metro error, Metro PSNR and MeshDev error for the rabbit normal mesh.
the unlifted butterfly wavelet transform, most of the tangential components are zero. Instead, if the Loop wavelet transform is used, there are more non-zero tangential components as shown in figure 5.10 (a).

Since the EQ coder was designed in such a way that all the tangential components are encoded using a single GGD with fixed shape and variance parameters at each scale, the EQ coder does not perform better than the zerotree coder in Loop wavelet transform case. Figure 5.10 (b) - (d) show the plots comparing the EQ and the zerotree coders using the Loop wavelet transform.

The zerotree coder gives the best results with the lifted butterfly wavelet transform. The same is the case with the EQ coder. Figure 5.11 shows plots comparing the different wavelet transforms for the EQ and zerotree coders for the Venus head normal mesh.

Figures 5.12 and 5.13 compare the EQ and the zerotree coders using the lifted butterfly wavelet transform, for the horse and the rabbit meshes respectively. Tables 5.2, 5.3 and 5.4 show the PSNR values obtained using the EQ and the zerotree coders with the different wavelet transforms for the Venus head, the horse and the rabbit meshes respectively.

For the horse mesh, there is not much difference in the performance of the EQ and the zerotree coders for the lifted butterfly wavelet transform case, whereas for the unlifted butterfly wavelet transform, the performance of the zerotree coder is 0 to 0.5 dB better than the EQ coder in terms of PSNR for the intermediate bits-per-vertex values. For the Venus head mesh, the gains in PSNR for the EQ coder are
Figure 5.10  Comparing the EQ and zerotree coders using the Loop and butterfly wavelet transforms. (a) Histograms of Loop and unlifted butterfly wavelet coefficients for the Venus head normal mesh at scale 4. (b)-(d) Metro error, Metro PSNR and MeshDev error for the Venus head normal mesh.
Figure 5.11 Comparing the different wavelet transforms for the EQ and zerotree coders. The Venus head normal mesh is used for these plots.
Figure 5.12 Comparing the EQ and zerotree coders using the lifted butterfly wavelet transform for the horse mesh.

Figure 5.13 Comparing the EQ and zerotree coders using the lifted butterfly wavelet transform for the rabbit mesh.
### Table 5.2  Venus head mesh PSNR comparison.

<table>
<thead>
<tr>
<th>bit rate</th>
<th>0.25</th>
<th>0.5</th>
<th>1.0</th>
<th>2.0</th>
<th>3.0</th>
<th>4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQ lifted BW</td>
<td>63.69</td>
<td>68.63</td>
<td>74.16</td>
<td>79.16</td>
<td>81.7</td>
<td>83.16</td>
</tr>
<tr>
<td>ZT lifted BW</td>
<td>62.95</td>
<td>68.21</td>
<td>73.66</td>
<td>78.85</td>
<td>81.66</td>
<td>81.92</td>
</tr>
<tr>
<td>EQ unfolded BW</td>
<td>63.48</td>
<td>68.59</td>
<td>74.08</td>
<td>78.85</td>
<td>81.37</td>
<td>82.96</td>
</tr>
<tr>
<td>ZT unfolded BW</td>
<td>62.4</td>
<td>67.78</td>
<td>73.0</td>
<td>78.4</td>
<td>81.2</td>
<td>81.5</td>
</tr>
<tr>
<td>EQ Loop</td>
<td>59.93</td>
<td>65.28</td>
<td>71.28</td>
<td>76.44</td>
<td>79.46</td>
<td>81.42</td>
</tr>
<tr>
<td>ZT Loop</td>
<td>60.88</td>
<td>66.1</td>
<td>71.81</td>
<td>77.07</td>
<td>79.71</td>
<td>79.72</td>
</tr>
</tbody>
</table>

### Table 5.3  Horse mesh PSNR comparison.

<table>
<thead>
<tr>
<th>bit rate</th>
<th>0.25</th>
<th>0.5</th>
<th>1.0</th>
<th>2.0</th>
<th>3.0</th>
<th>4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQ lifted BW</td>
<td>61.96</td>
<td>69.56</td>
<td>76.35</td>
<td>80.85</td>
<td>82.7</td>
<td>83.79</td>
</tr>
<tr>
<td>ZT lifted BW</td>
<td>61.92</td>
<td>69.14</td>
<td>76.07</td>
<td>80.63</td>
<td>82.37</td>
<td>83.91</td>
</tr>
<tr>
<td>EQ unfolded BW</td>
<td>62.82</td>
<td>69.18</td>
<td>76.07</td>
<td>80.96</td>
<td>82.84</td>
<td>83.81</td>
</tr>
<tr>
<td>ZT unfolded BW</td>
<td>62.73</td>
<td>69.87</td>
<td>76.53</td>
<td>81.22</td>
<td>82.82</td>
<td>84.19</td>
</tr>
</tbody>
</table>

### Table 5.4  Rabbit mesh PSNR comparison.

<table>
<thead>
<tr>
<th>bit rate</th>
<th>0.25</th>
<th>0.5</th>
<th>1.0</th>
<th>2.0</th>
<th>2.5</th>
<th>3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQ lifted BW</td>
<td>70.25</td>
<td>75.72</td>
<td>80.9</td>
<td>84.21</td>
<td>85.06</td>
<td>85.55</td>
</tr>
<tr>
<td>ZT lifted BW</td>
<td>69.31</td>
<td>75.12</td>
<td>80.91</td>
<td>84.0</td>
<td>84.07</td>
<td>84.07</td>
</tr>
<tr>
<td>EQ unfolded BW</td>
<td>69.97</td>
<td>75.29</td>
<td>80.57</td>
<td>83.96</td>
<td>84.96</td>
<td>85.54</td>
</tr>
<tr>
<td>ZT unfolded BW</td>
<td>68.69</td>
<td>74.65</td>
<td>80.42</td>
<td>83.56</td>
<td>83.61</td>
<td>83.61</td>
</tr>
</tbody>
</table>
between 0.5 to 1 dB for the lifted butterfly wavelet transform case, around 1 dB for the unlifted butterfly wavelet transform case. The zerotree coder performs better at lower bit-rates for the Venus head mesh with the Loop wavelet transform. For the rabbit mesh, the PSNR gains for the EQ coder are between 0 to 1 dB for the lifted butterfly wavelet transform case and the gains are between 0 to 1 dB for the unlifted butterfly wavelet transform case.

5.5.4 Reconstructed meshes at different bit-rates

Figure 5.14 shows the Venus head normal mesh reconstructions at different bits-per-vertex values using the EQ coder. Note that there is not much visible improvement at higher bits-per-vertex values. The difference between the meshes at the highest bits-per-vertex values are mainly localized in the non-smooth regions like the regions around the eyes.

5.5.5 Progressive nature of the coders

The zerotree coder is progressive in nature, whereas the EQ coder is not. In the zerotree coder, the most significant bit planes are sent first and the mesh is progressively decoded. However, in the EQ coder, the mesh can be decoded progressively only in terms of level. In other words, we can decode the coefficients at each scale starting with the coarser scales. Figure 5.15 compares the progressive nature of the EQ and the zerotree coders. The error function resembles a stair-case function in the EQ case, where each step corresponds to a new finer level.
Figure 5.14  The Venus head normal mesh reconstruction using EQ coder at different bits-per-vertex (bpv).
Figure 5.15  Comparing the progressive nature of the EQ and zerotree
coders using the Metro error and PSNR for the Venus head mesh. EQ 1, EQ
2 and EQ 3 correspond to $\lambda = 10^{-4}$, $\lambda = 10^{-6}$ and $\lambda = 10^{-8}$ respectively, where $\lambda$ is the rate distortion parameter used for encoding the mesh with the
EQ coder.
The staircase functions in Figure 5.15 correspond to the meshes decoded by the EQ coder scale-wise. The Venus head normal mesh has 6 scales of wavelet coefficients.
Chapter 6

Discussion and Conclusions

6.1 Conclusions

In this work, we presented a context based coder for normal meshes and compared it with the zerotree coder for the normal meshes. The wavelet coefficients are modeled using a GGD model with a slowly spatially-varying variance field. An appropriate quantizer is chosen based on R-D optimization. The quantized symbols are encoded using an arithmetic coder. We also introduced an ordering and neighborhood definitions for the vertices used in the EQ algorithm. The scaling coefficients are encoded using uniform quantization and the base mesh connectivity is encoded using a TG coder. Metro and MeshDev error tools are used to compare the performance of the EQ coder with that of the zerotree coder. We observe performance gains of around 0.5 to 1 dB for the EQ coder. The performance of the EQ coder is close to the upper bounds obtained using the complete neighborhood information with the original coefficient values.

6.2 Future work

We hope to improve the error metrics used for the R-D optimizations and replace the MSE with a vertex-based error metric. We hope to get better results as the final global error measurement would better correspond to a vertex-based error metric.
compared to the MSE.

We hope to study the R-D trade-offs between the scaling coefficients, the normal wavelet coefficients and the tangential wavelet coefficients. Empirical results suggest that the error contribution from the scaling coefficients is much larger compared to that of the wavelet coefficients. For smooth surfaces, we observe that the tangential components do not contribute significantly to the global error. We propose to study the effects of quantization errors of different components of the wavelet coefficients on the global error measurements.

We also intend to implement the zerotree-based Space-Frequency Quantization (SFQ) algorithm [30] for normal meshes and compare the performance with the EQ and the zerotree coders for normal meshes. We hope to get better results with the SFQ algorithm as it uses R-D optimized zerotree coding.
Bibliography


Appendix A

Asymmetric Hausdorff Distance

The Hausdorff distance is used in section 5.1 to define error metrics to compare two meshes describing the same surface. The Hausdorff distance between two sets $A$ and $B$ is defined as the

$$h(A, B) = \max_{a \in A} \{ \min_{b \in B} \{d(a, b)\} \}$$

where $d(a, b)$ is any metric distance between the points $a$ and $b$. Figure A.1 shows the asymmetric nature of the Hausdorff distance defined above.

![Diagram](image)

$A = \{a_1, a_2\}, B = \{b_1, b_2, b_3\}$

$$h(A, B) = d(a_1, b_1)$$

$$h(B, A) = d(b_3, a_2)$$

$$H(A, B) = \max \{ h(A, B), h(B, A) \}$$

**Figure A.1** Asymmetric nature of Hausdorff distance.

A symmetric Hausdorff distance, $H(A, B)$, can be defined using the maximum of the $h(A, B)$ and $h(B, A)$ as shown below.

$$H(A, B) = \max \{ h(A, B), h(B, A) \}$$
Appendix B

Metro and MeshDev Sample Output

B.1 Metro sample output

A sample output of the Metro error tool, obtained by comparing the original irregular Venus head mesh and an approximation of the Venus head mesh is shown below. The software program accepts input meshes in the Open Inventor format. The Metro output includes the number of vertices and triangles in the two input meshes, the bounding box diagonal length. The error output values includes the mean error and the mean squared error in absolute terms and as a percentage of the bounding box diagonal.

Metro, Version 2.5 (10-98)

Visual Computing Group

I.E.I.-CNR and CNUCE-CNR, Pisa (Italy)

<table>
<thead>
<tr>
<th></th>
<th>Mesh1</th>
<th>Mesh2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orientable</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>2Manifold</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Closed</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Vertices</td>
<td>50002</td>
<td>163842</td>
</tr>
<tr>
<td>Triangles</td>
<td>100000</td>
<td>327680</td>
</tr>
<tr>
<td>Conn. Comp</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BBox Diag _</td>
<td>1.5647</td>
<td>1.5552</td>
</tr>
<tr>
<td>Diameter _ _</td>
<td>1.1109</td>
<td>1.0883</td>
</tr>
<tr>
<td>Edge Lengt</td>
<td>7.0279</td>
<td>0</td>
</tr>
<tr>
<td>Area ______</td>
<td>2.3518</td>
<td>2.3048</td>
</tr>
<tr>
<td>Volume _____</td>
<td>0.2786</td>
<td>0.2786</td>
</tr>
</tbody>
</table>

Scan 1:
CPU Time: 235.36 secs

Scan 2:

| Mesh1 | Mesh2 |
| Net    | 104x 72x104 | 155x108x155 |
| Items. | 314018 | 863485 |
| Ite x face. | 3.14018 | 2.63515 |
| Grid full.. | 4.65193% | 3.06852% |
| Samples.... | 11975040 |

<table>
<thead>
<tr>
<th>(Value)</th>
<th>(BBox %)</th>
<th>(Diam. %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01599</td>
<td>(1.0223%)</td>
<td>(1.44%)</td>
</tr>
<tr>
<td>0.01717</td>
<td>(1.0976%)</td>
<td>(1.546%)</td>
</tr>
</tbody>
</table>

| E+ .. | 0.002318 | (0.1481%) | (0.2087%) |
| E- .. | 0.002387 | (0.1525%) | (0.2148%) |
| Et .. | 0.002353 | (0.1503%) | (0.2118%) |
--------------- Mean Square Error -------------

E+ ...: 0.003002 (0.1919%) (0.2703%)
E- ...: 0.003128 (0.1999%) (0.2816%)
Et ...: 0.003067 (0.196%) (0.276%)

--------------- Volume of Difference -------------

V+ ...: 31471.7
V- ...: 34169.4
Vt ...: 131234

B.2 MeshDev sample output

A sample output of the MeshDev tool used to compare the original irregular horse mesh and an approximation of the horse mesh is shown below. The software program accepts input meshes in the SMF format. The MeshDev output includes the number of vertices and triangles in the input meshes and the mean geometric deviation between the two input meshes.

```
MeshDev -d 0 horse1.smf horse2.smf

MeshDev 0.3
Mesh Comparison Software
Copyright (C) 2002 Michael ROY

Model load... 7.862s
```
Initialization... 0.480s
Processing... 36.272s
Total time... 44.614s

Mesh 1
horse1.smf
Vertices: 112642
Faces: 225280
Area: 0.0357

Mesh 2
horse2.smf
Vertices: 48485
Faces: 96966
Area: 0.035891

Geometric Deviation
Minimum: 2.03703e-010
Maximum: 0.00158073
Mean: 0.000161311
Variance: 1.90336e-008