3D MODEL COMPRESSION USING IMAGE COMPRESSION BASED METHODS

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND

ELECTRONICS ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Kıvanç Köse

January 2007

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Enis Çetin(Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Levent Onural

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Uğur Güdükbay

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Approved for the Institute of Engineering and Sciences:

Prof. Dr. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

3D MODEL COMPRESSION USING IMAGE COMPRESSION BASED METHODS

Kıvanç Köse
M.S. in Electrical and Electronics Engineering
Supervisor: Prof. Dr. Enis Çetin
January 2007

A connectivity-guided adaptive wavelet transform based mesh compression algorithm is proposed. On the contrary to previous work, which process the mesh models as 3D signals, the proposed method uses 2D image processing tools for compressing the mesh models. The 3D mesh is first transformed to 2D images on a regular grid structure by performing orthogonal projections onto the image plane. This operation is computationally simpler than parameterization. The neighborhood concept in projection images is different from 2D images because two connected vertex can be projected to isolated pixels. Connectivity data of the 3D mesh defines the interpixel correlations in the projection image. Thus, the wavelet transforms used in image processing do not give good results on this representation. Connectivity-Guided Adaptive Wavelet Transform is defined to take advantage of interpixel correlations in the image-like representation. Using the proposed transform the pixels in the detail (high) subbands are predicted from their connected neighbors in the low-pass (lower) subbands of the wavelet transform. The resulting wavelet data is encoded using either "Set Partitioning In Hierarchical Trees" (SPIHT) or JPEG2000. SPIHT approach is progressive because different resolutions of the mesh can be reconstructed from different partitions of SPIHT bitstream. On the other hand, JPEG2000 approach is a single rate

coder. The quantization of the wavelet coefficients determines the quality of the reconstructed model in JPEG2000 approach. Simulations using different basis functions show that lazy wavelet basis gives better results. The results are improved using the Connectivity-Guided Adaptive Wavelet Transform with lazy wavelet filterbanks. SPIHT based algorithm is observed to be superior to JPEG2000 and MPEG in rate-distortion. Better rate distortion can be achieved by using a better projection scheme.

Keywords: 3D Model Compression, Image-like mesh representation, Connectivity-guided Adaptive Wavelet Transform

ÖZET

ÜÇ BOYUTLU MODELLERİN İMGE SIKIŞTIRMA YÖNTEMLERİYLE SIKIŞTIRILMASI

Kıvanç Köse
Elektrik ve Elektronik Mühendisliği Bölümü, Yüksek Lisans
Tez Yöneticisi: Assoc. Prof. Dr. Enis Çetin
Ocak 2007

Üç boyutlu modellerin imge sıkıştırma yöntemleri kullanılarak sıkıştırılması için bir yöntem önerilmektedir. Önerilen yöntem, literatürdeki birçok algoritmanın tersine, modellerin 3-Boyutlu veriler yerine 2-Boyutlu veriler olarak ele almaktadır. 3-Boyutlu modeller ilk olarak düzenli ızgara yapıları üzerinde 2-Boyutlu imgelere dönüştürülmektedir. Önerilen yöntem diğer yöntemlerde kullanılan parametrizasyon tekniğine göre, hesaplama açısından daha basittir. Elde edilen imge benzeri temsilin sıradan imgelerden tek farkı, pikseller arası ilintinin yanyanalık ile değil, 3-boyutlu modelin bağlanılırlık verisi kullanılarak sağlanmasıdır. Bu nedenle yaygın kullanılan dalgacık dönüşümü teknikleri bu temsil üzerinde çok iyi sonuçlar vermemektedir. Burada önerilen Bağlanılırlık Bazlı Uyarlamalı Dalgacık Dönüşümü sayesinde dalgacık dönüşümü sıradüzensel yapısının detay katmanlarında bulunan piksel değerleri, alçak frekans katmanlarında bulunan komşularından öngörülebilmektedir. Böylece oluşturulan dalgacık dönüşümü verileri Sıradüzensel Ağaç Yapılarının Kümelere Bölüntülenmesi (Set Partitioning In Hierarchical Trees - SPIHT) ya da JPEG200 tekniklerinden biri kullanılarak kodlanmaktadı. SPIHT tekniği sayesinde elde edilen veri dizgisi aşamalı gösterime uygundur; çünkü, dizginin farklı uzunluktaki bölümlerinden farklı çözünürlüklerde modeller geri çatılabilmektedir.

JPEG200 yönteminin burada önerilen şekli tek çözünürlüklü geriçatıma olanak sağlamaktadır. Önerilen yöntemde dalgacık dönüşümü katsayılarının nicemlenme şekli geri çatılan modelin çözünürlüğünü belirlemektedir. Farklı dalgacık dönüşümü taban vektörleri kullanılarak yapılan deneyler sonucunda lazy dalgacık dönüşümünün en iyi sonuçları verdiği gözlemlenmiştir. Bağlanırlık bazlı uyarlamalı dalgacık dönüşümü kullanılarak yapılan deneylerin sonuçlarında bir önceki yönteme göre gelişme gözlemlenmiştir. Dalgacık dönüşümü verilerinin SPIHT ile kodlanmasıyla elde edilen sonuç, JPEG2000 ile yapılan kodlamanın sonucundan ve 3B modellerin MPEG ile kodlamasından daha başarılı olmuştur. Daha iyi sonuçlar elde etmek için daha iyi bir izdüşüm methodu denenmelidir.

Anahtar kelimeler: 3 Boyutlu modellerin Sıkıştırılması, 3B modellerin imge benzeri temsili, Bağlanılırlık BazlıUyarlamalıDalgacık Dönüşümü

ACKNOWLEDGEMENTS

I gratefully thank my supervisor Prof. Dr. Enis Çetin for his supervision, guidance, and suggestions throughout the development of this thesis.

I also would like to thank Prof. Dr. Uğur Güdükbay and Prof. Dr. Levent Onural for reading and commenting on the thesis.

This work is supported by the European Commission Sixth Framework Program with Grant No: 511568 (3DTV Network of Excellence Project).

Contents

List of Figures

List of Tables

To My Family and most beloved...

Chapter 1

Introduction

The demand to visualize the real world scenes in digital environments and make simulations using those data ~~is~~ increased in last years. Three-dimensional (3D) meshes are used for representing 3D objects. The mesh representations of 3D objects are created either manually or by using 3D scanning and acquisition techniques [1]. Meshes with arbitrary topology can be created using these methods. The 3D geometric data is generally represented ~~using~~ two tables: *a vertex list* storing the 3D coordinates of vertices and *a polygon list* storing the connectivity of the vertices. The polygon list contains pointers into the vertex list.

Multiresolution representations can be defined for 3D meshes [2]. In a fine resolution version of an object, more vertices and polygons are used as compared to a coarse representation of the object. It would be desirable to obtain the coarse representation from the fine representation using computationally efficient algorithms. Wavelet-based approaches are applied to meshes to realize a multiresolution representation of a given 3D object [2].

As the scenes and the objects composing those scenes becomes more complex and detailed, the size of the data also grows. So the problem of transmitting this data from one place to another becomes a more difficult and important task.

The transmission can be over a band limited channel, either from one system to another system or from a storage device to processing unit eg. from *main memory* to *graphics card* [3].

"The fundamental problem in communication is that of reproducing at one point either exactly or approximately a message selected at another point" [4]. However the choice of the message data is not unique. There exists several possible sets of messages that can be used to describe the transmitted information. The problem is creating a description that expresses the data best with the smallest size.

There exists several mesh compression approaches in the literature. Most of those approaches treats the meshes as 3D graphs in the space. The geometry and the connectivity data is compressed separately. The *geometry images* technique explained in [5] compresses the meshes using image compression methods. The meshes are parameterized to two dimensional (2D) planes. Those parameterizations of meshes are treated as images and compressed using a wavelet based image coder. Parameterization of a surface mesh is a complex task to be applied to an arbitrary object because many linear equations need to be solved. As the objects get more complex, ~~the~~ parameterization ~~operation~~ becomes nearly impossible. The surfaces are cut for reducing the complexity of parameterization [6]. The adaptation of signal processing algorithms [7] to surface meshes is also a challenging task although it is easier than parameterization. It is much easier to transform the data and apply any algorithm that is needed rather than adapting signal processing algorithms to 3D graphs.

These drawbacks in [5] gave us the idea of finding easier ways for mapping meshes to images and using image compression tools directly on those images. Since image processing is a well established branch of signal processing, there exists a wide spectrum of algorithms that can be used. Thus, understanding of

the fundamentals behind compression especially image compression is an essential issue in our work.

1.1 Mesh Representation

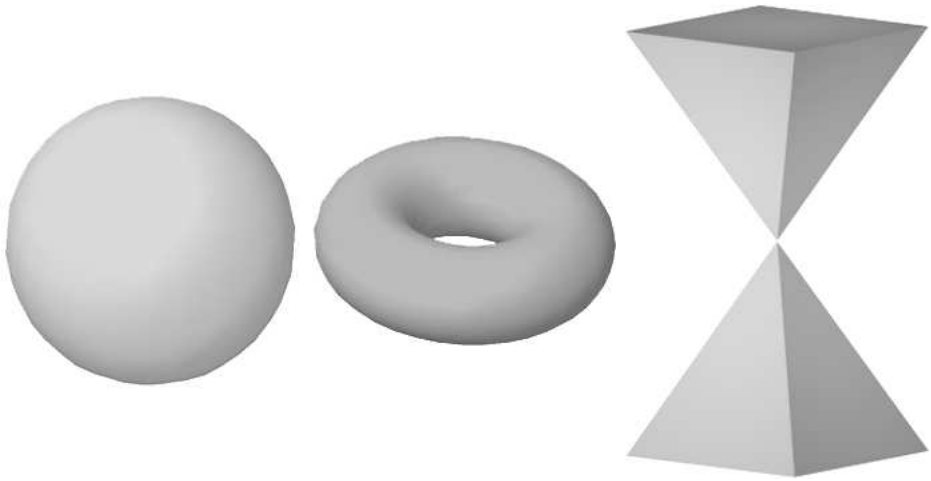
3D Meshes are visualization of 3D objects using vertices *(geometry)*, edges, faces, some attributes like surface normals, texture, color, etc, and *connectivity*. 3D points $\{\mathbf{v_1}, ..., \mathbf{v_n}\} \in \mathbf{V}$ *in* $\mathbf{R}^3$ are called *vertices* of a 3D mesh. The *convex hull* of two vertices in $\mathbf{R}^3$, $conv\{\mathbf{v_n}, \mathbf{v_m}\}$ is called an *edge*. So an edge is mapped to line segment in $\mathbf{R}^3$ with end points at $\mathbf{v_n}$ and $\mathbf{v_m}$. *Face* of a triangular mesh is a surface which is $conv\{\mathbf{v_n}, \mathbf{v_m}, \mathbf{v_k}\}$. Thus, a face is mapped to a surface in $\mathbf{R}^3$ that is enclosed by the edges incident to the vertices $\mathbf{v_n}, \mathbf{v_m}, \mathbf{v_k}$. A face may have no direction or its direction can be determined using the surface normals data. The additional attributes of a mesh are mostly carried by the vertices. That information can be extended along the edges and the faces using linear interpolation or other techniques (eg. linear, phong shading of a surface).

The connectivity information summarizes which mesh elements are connected to each other. Edges $\{\mathbf{e_1}, ..., \mathbf{e_n}\} \in \mathbf{E}$ are incident to its two end vertices. Faces $\{\mathbf{f_1}, ..., \mathbf{f_n}\} \in \mathbf{F}$ are surrounded by its composing edges and incident to all the vertices of its incident edges. The edges have no direction. Two types of mesh connectivity are common in mesh representations. *Edge Connectivity* is the list of edges in the mesh and *Face Connectivity* list of faces in the mesh. In a triangular mesh, since all the vertices ~~incident to~~ a face lie in a plane, the face also lies in a plane. In polygonal meshes the number of the vertices that are incident to the face, is four or more. So face of a polygonal mesh not necessarily lie in a plane.

Vertices of a mesh can be incident to any number of edges. The number of the edges that are incident to a vertex is ~~named as~~ the *valence* of the vertex [8].

The number of the edges that are incident to a face is ~~named as~~ the *degree* of a face [8].

The number of ~~the~~ faces incident to an edge and number of ~~the~~ face loops incident to a vertex, are important concepts while defining, if the mesh is manifold or non-manifold. "A *2-manifold* is a topological surface where every point on the surface has a neighborhood topologically equivalent to an open disk of $\mathbf{R}^2$" [9]. If the neighborhood of a point on the surface is equivalent to an half disk ~~than~~ the mesh is *manifold with boundary* [9]. In Figure 1.1 examples of manifold ~~and~~ manifold with boundary and non-manifold surfaces can be seen.



(a) (b) (c)

Figure 1.1: Sphere (a) and torus (b) are manifold surfaces. ~~The connection of two rectangles edge to edge,~~ as seen in (c), creates a non-manifold surface. Sphere has no hole so it is a genus-0 surface, where torus has one hole so it is a genus-1 surface. All the objects have one shell.

Two other important concept about meshes are *shell* and *genus*. Shell is a part of the mesh that is edge-connected. The genus of a mesh is an integer that can be derived from the number of closed curves that can be drawn on the mesh without dividing it into two or more separate pieces. It is equal to the number of handles on the mesh object [8]. As seen in Figure 1.1(b) a torus is genus-1 since it has one hole and sphere Figure 1.1(a) is genus-0 since it has no hole [8].

Connectivity of a mesh is a quadruple $(\mathbf{V}, \mathbf{E}, \mathbf{F}, \mathbf{Q})$ where $\mathbf{Q}$ is the incidence relation [9]. Relationships between mesh elements can be found using the connectivity information in Euler equation [8]. The Euler characteristics $\varkappa$ of a mesh can be calculated using :

$$\varkappa = v - e + f, \tag{1.1}$$

where v, e, f are the number of vertices, edges and the faces of a mesh respectively. The Euler characteristics of a mesh depends on the number of shells, genus and boundaries of the mesh. For closed and manifold meshes, the Euler characteristic is given as :

$$\varkappa = 2(s - g), \tag{1.2}$$

where s is the number of the shells and g is the genus number.

Simple meshes are meshes that are homeomorphic to a sphere which means topologically they are the same. Homeomorphism is a function between two spaces which satisfies the conditions of having bijection, being continuous and having a continuous inverse [10]. In other words, homeomorphism is a continuous stretching and bending of the mesh into a new shape.

Each triangle of a simple mesh has exactly three edges and each edge is adjacent to exactly two triangles. This leads us to :

$$2e = 3f. \tag{1.3}$$

Substituting Equation 1.3 in to Equation 1.1 we obtain :

$$v - e + f = 2, \tag{1.4}$$

$$v - \frac{1}{2}f = 2, \tag{1.5}$$

$$\frac{v}{f} = \frac{2}{f} + \frac{1}{2}. \tag{1.6}$$

For large meshes the assumption of Equation 1.7 can be made. Considering that each edge is connected to its two end vertices, the average valence for large, simple meshes can be calculated by averaging the sum of the valence $val_{\mathbf{i}}$ of each individual vertex $\mathbf{v_i}$ where $\mathbf{i} \in \mathbf{Z}$, as in :

$$f = 2v \; and \; e = 3v, \tag{1.7}$$

$$\frac{1}{v}\sum_{\mathbf{i}\,=\,1}^{v} val_{\mathbf{i}} \approx \frac{2e}{v} = 6. \tag{1.8}$$

Basically a mesh is a pair $\mathbf{M} = (\mathbf{C}, \mathbf{G})$ where $\mathbf{C}$ represents the mesh connectivity information and $\mathbf{G}$ represents the geometry (3D coordinates). Connectivity information is closely related with the mesh elements whose adjacency and incidence informations are important for navigating in the mesh. Different mesh manipulation and compression algorithms depend on different properties of the mesh. Therefore ~~representing~~ the mesh in an appropriate way so that it can be easily used by those algorithms ~~is also essential~~.

For example, a connectivity compression algorithm *EdgeBreaker* [11], traverses faces of the mesh to code the connectivity data of the mesh. Therefore usage of a data structure which enables easy access to adjacency information of the mesh, would increase the speed of the algorithm. Thus, choosing an appropriate data structure, ~~to be~~ used with the algorithms~~s~~ is also an important performance issue. Different edge based data structures such as *halfedge data structure* [12] and *winged edge data structure* [13] are discussed in [14]

1.2 Compression

Compression is selection of a more convenient description for transmitting the information using the smallest data size. For lossless transmission the best that can be done is to reach a near entropy bit-rate. For lossy transmission the situation is more complex. For different distortion levels of the data, different entropies can be found. Thus different lower bounds for data compression exists. The idea is to transmit the information using smallest amount of information size by which the original data can be reconstructed in the predefined distortion bounds.

Mostly data and information are confused with each other. Data is the means by which the information is conveyed. However various data sizes can be used to transmit the same amount of information. It is just like two person describing the same scene in two different ways. One may tell it is a "forest" and the other may tell "several trees enumerated near each other". *Data compression* is the process of reducing the data size required to represent a given quantity of information [15]. For example a 256 color leveled 1024 x 1024 sized image which has linearly independent uniformly distributed pixel color values, would have a data size of 8 MB. However, if the image has only one color, it can be coded using 8 bits for its color information and 20 bits for its size information which is a total of nearly 3 Bytes. For this image, the remaining data restates the already known information that can be named as data redundancy.

1.2.1 Compression and Redundancy

There exists several types of redundancies for different types of data. For example in digital image compression, three basic types of redundancy are *coding, interpixel and pyschovisual* redundancy. For the proposed 2D representations of

meshes all these redundancies are valid except interpixel redundancy which left its place to redundancies between vertex positions (geometry) and connectivity.

Explanation of coding redundancy comes from the probability theory. If a symbol is more probable than another one, it should be represented using less number of bits. Histograms are very useful for this purpose since they show how many times a value is taken in the data. Normalized versions of histograms with infinite number of samples converges to probability density functions with the ergodicity assumption. So histograms with finite number of samples gives a very good insight about the probability of a specific symbol. Probability of each level in an L color image with n pixel is :

$$p(k) = \frac{n_k}{n}, \ k = 0, 1, ..., L-1, \tag{1.9}$$

where n_k is the number of times that k^{th} gray level appears in the image, and n is the number of image pixels. The Average Code Length (ACL) can be calculated using :

$$ACL = \sum_{k=0}^{L-1} l(k)p(k), \tag{1.10}$$

where $l(k)$ is the code length of the level k.

To minimize the total code length, average code length must be minimized. Using different symbol code lengths and giving shorter codes to the symbols with higher probability will minimize the average code length.

Lower variance in the histogram leads to better compression as the data is more predictable. The histogram of a single colored image has only one value at one specific color level so the whole data can be predicted from that pixel very well. Another example can be coding sentences in English. The *'e'* letter is the

most frequently used letter in English so assigning it a shorter symbol than the other letters, gives us smaller coded data size.

Pyschovisual redundancy comes from the imperfection of the human visual system. The difference between the positions of two points can not be perceived after some distance. So quantizing coordinates of points in 3D space do not change the visualization of the data.

In images neighboring pixels are not mostly independent from each other. This means there is a correlation between them. At the low frequency parts of the image this correlation is even more stronger. This is also true for meshes. But the situation is a little bit different. The neighboring concept is dependent on not only being near to each other but also being connected to each other. Connected pixels can be predicted from each other more reliably. Like the images, at the low pass parts of the object (smooth parts of the object) this correlation is more stronger.

1.3 Related Work on Mesh Compression

There are several mesh compression algorithms in the literature [16, 8, 17]. They can be classified as *single-rate compression* and *progressive mesh compression*. This classification can be further extended to sub groups called, *connectivity* and *geometry* compression. In most of the algorithms the connectivity information is exploited to more efficiently compress the geometry information of the mesh.

Compression is not the only way of reducing the size of a mesh. Methods like simplification and remeshing are also used for this purpose. In this section those concepts will also be mentioned in the context of compression. The aim of compression that is mentioned here is to transmit a mesh from one place to another using as small information as possible. To achieve this aim, connectivity

is encoded in a lossless manner and geometry is encoded in either lossless or lossy manner. Due to the quantization of the 3D positions of the vertex points, most of the geometry encoders are lossy.

Remeshing is a popular method for converting an irregular mesh to a semi-regular or regular mesh. The idea in remeshing is to represent the same object with more correlated connectivity and vertex position information while causing the least distortion. This makes the new mesh more compressible since many of the vertices can be predicted from its neighbors. It can be thought of as regularizing the mesh data. However, remeshing is a complex task.

Single rate encoders compress the whole mesh into a single bit stream. The encoded connectivity and the geometry streams are meaningless unless the whole stream is received by the decoder. In the context of single rate encoders, there exists several algorithms like, triangle strip coding [3], topological surgery [18], EdgeBreaker [11], TG coder [19], valence-based coder of Alliez and Desbrun [16], Spectral coding [20], geometry images [5]. They are also used in compression of the base meshes of progressive representation. Single rate mesh compression will be dealt in more detail at 1.3.1

Progressive compression has the advantage of representing the mesh at multiple detail levels. A simplified version of the original model can be reconstructed from some initial chunks of coded stream. The remaining chunks will add refinements to the reconstructed model This is an important opportunity for environments like Internet or scenes with multiple meshes where impression of existence of the object is more important then its details.

Progressive mesh representations [21] store a 3D object as a coarse mesh and a set of vertex split operations that can be used to obtain the original mesh. Progressive mesh representations use simplification techniques such as edge collapse

and vertex removal to obtain the coarser versions of the original mesh. Subdivision technique can also be regarded as a progressive mesh representation [22, 46]. In [24, 2], multiresolution approaches for mesh compression using wavelets are developed. Progressive mesh compression will be dealt in more detail at 1.3.2

Besides static meshes, there exists animation of meshes called dynamic meshes. Some of the static and progressive mesh compression algorithms are used for encoding them. Besides intra-mesh redundancies those mesh sequences have inter-mesh redundancies. Each mesh can be thought as a frame of an animation and the redundancies between frames should be exploited for more efficiently compressing dynamic meshes. Several algorithms like, Dynapack [25], D3DMC [26, 27], RD-D3DMC [28, 27] exists for encoding dynamic meshes.

Uncompressed mesh data is large in size and contains redundant information. Each triangle is specified by three vertices which are 3D points in $\mathbf{R}^3$ and some attributes like surface normals,color,etc. If the vertex coordinates are quantized to 4 byte floating point values, a vertex needs 12 bytes only for its geometry information. From equation 1.8 it is known, that each vertex is connected to 6 triangles. If a triangle is represented by the data of its three vertices, each vertex would be transmitted 6 times. This means a data flow of 72 bytes per vertex. To decrease this data flow, more efficient representations of the mesh should be used. Encoding the mesh is one of the best way of reducing this data flow.

1.3.1 Single Rate Compression

A 3D mesh model is basically composed of two data namely, *Connectivity* and *Geometry*. So two types of compression for those two data is given; *Connectivity Comprssion* and *Geometry Compression*.

Connectivity Compression

In the connectivity information each vertex is six times repeated in average. One simple way to reduce the number of vertex transmission is creating triangular strips from the mesh (Figure 1.2). In this method, the initial triangle is described using its three vertices. The next triangle which is a neighbor of the current one, will be formed using the two vertices from the joined edge and a new vertex. So for each triangle of the strip, one vertex will be transmitted. From Equation 1.7 it is known that in large simple meshes there exist twice as many faces as the vertices. So if triangle strip method is used, each vertex will be transmitted twice.

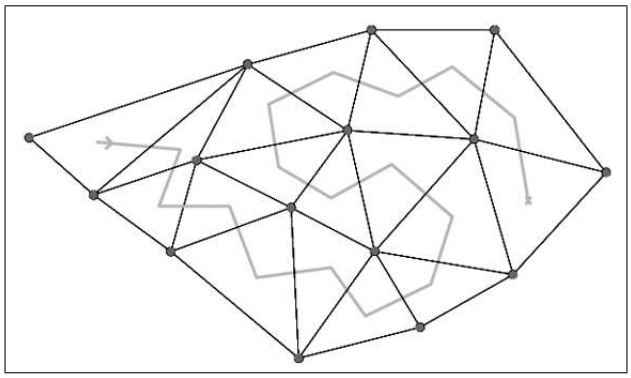
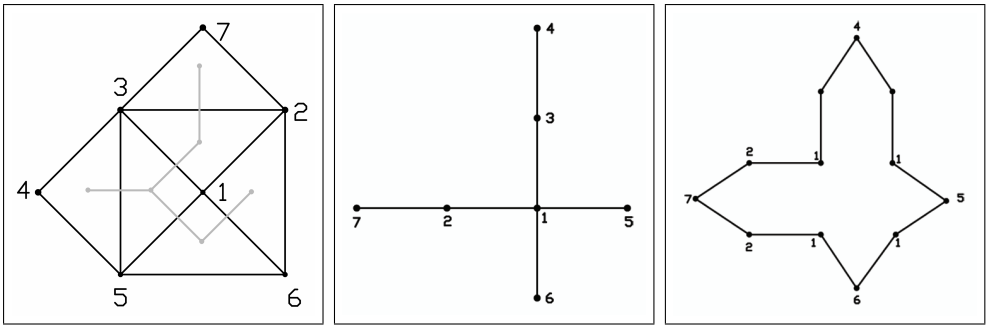


Figure 1.2: A triangle strip created from a mesh.

Deering [3] proposed an idea of using a vertex buffer to store some of the transmitted vertices in order not to transmit each vertex twice. The new triangle in the strip either introduces a new vertex which will be pushed in the vertex buffer or a reference to the vertex buffer. This gives the algorithm the advantage of re-using the transmitted vertices.

For achieving this, Deering proposed a new mesh representation called *generalized triangle mesh*. The mesh is converted into an efficient linear form using which the geometry can be extracted by a single monotonic scan over the vertex array. However references to used vertices are inevitable. Buffering solves the problem to some extend. Deering used a vertex buffer with size 16.

A triangle strip is a part of triangle spanning tree (Figure 1.3(a)) whose nodes are the faces of the mesh and edges are the adjacent edges of the mesh faces. This is the dual of the connectivity graph whose nodes are the vertices (Figure 1.3(b)) and edges are the edges of the mesh. Taubin proposed *Topological Surgery* [18] algorithm to encode the connectivity of a mesh using those two trees. Both the triangle and the vertex spanning trees are encoded and sent to the receiver. The decoder at the receiver first decompresses the vertex spanning tree and doubles each of its edges (Figure 1.3(c)). After decoding the triangle spanning tree the resulting triangles are inserted between pairs of doubled edges of the vertex spanning tree.



(a) (b) (c)

Figure 1.3: Spanning trees in the triangular mesh strip method.

Region growing is another approach for connectivity coding. Rossignac's EdgeBreaker [11] is one of the examples for this approach. In [11] a region growing starting with a single triangle is done. The grown region contains the processed triangles. The edges between the processed and to be processed region is called the *cut-border*. A selected edge in the cut border which defines the *next processed triangle*, is called the gate. Those mentioned elements can be seen in Figure 1.4.

Each next triangle can have among five possible different orientations with respect to the gate and cut-border. Each orientation has its own opcode. By this way each processed triangle is associated with an opcode. This iterative process goes on until no unprocessed triangles left in the mesh. Figure 1.5

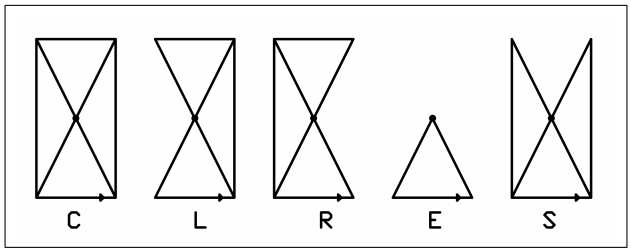


Figure 1.4: Opcodes defined in the EdgeBreaker.

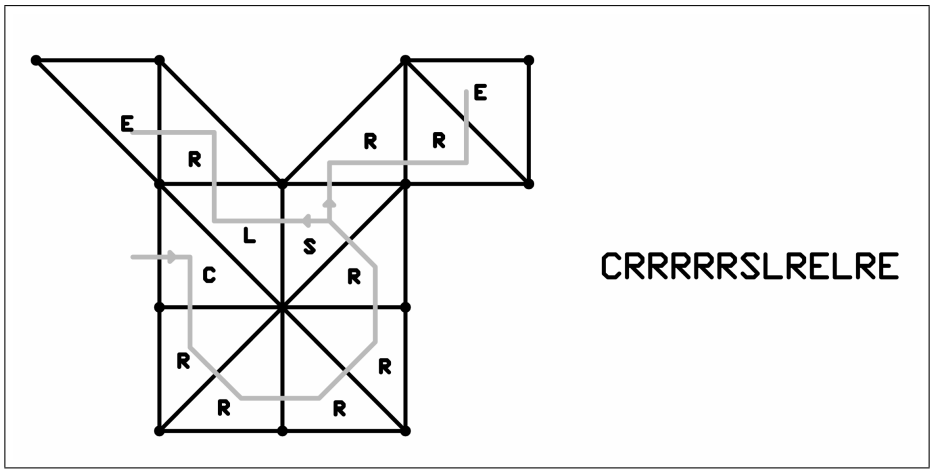


Figure 1.5: Encoding example for EdgeBreaker.

Since the decoder knows how the encoder coded the mesh connectivity, it can reconstruct the data by inverting the operations done by the encoder. Figure 1.6 shows decoding of the mesh in Figure 1.5. *Split* operation is a special occasion while decoding. Each *split* operation has an associated *end* operation which finishes the cut-border introduced by the *split* operation. The length of each run starting with a *split* is needed while decoding the symbol array. Therefore EdgeBreaker needs two runs over the symbol array.

This problem is solved by *Wrap&Zip* [29] and *Spiral Reversi* [30] which are extensions on EdgeBreaker. Wrap&Zip solves the problem by creating dummy vertices for *split* operations while encoding the mesh (wrap) and identifying them in the decoding phase (zip). Spiral Reversi decodes the symbol stream starting from the end. By this way the decoder first finds the *end* opcodes and then associated *split* opcodes.

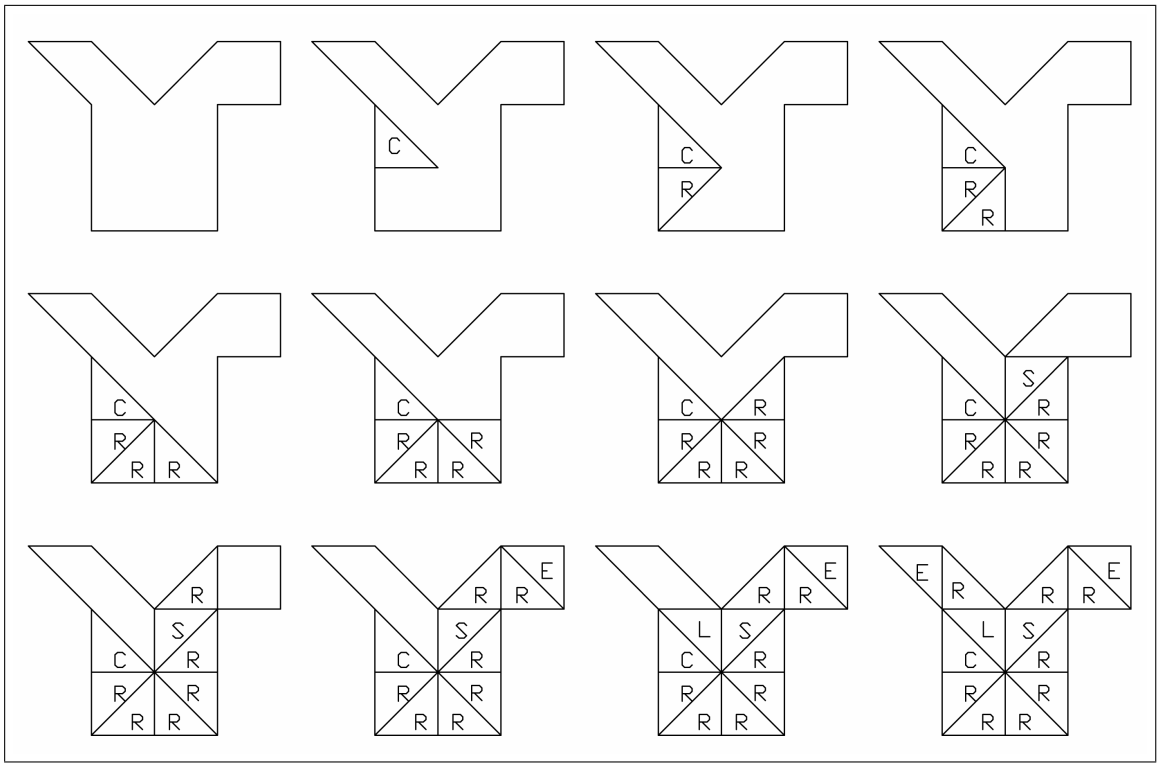
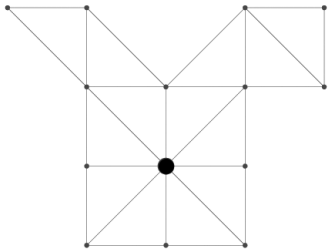


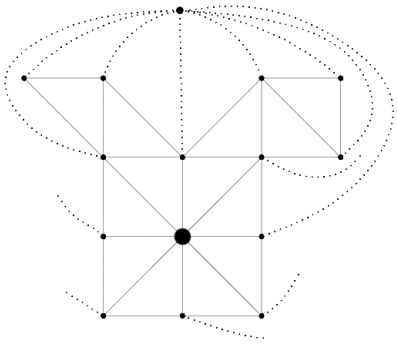
Figure 1.6: Decoding example for EdgeBreaker.

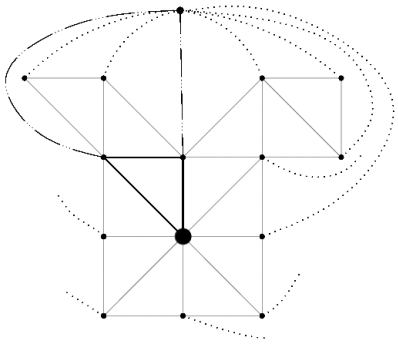
In [19], Touman and Gotsman also introduced a region growing algorithm. But instead of coding the relation between gates and the triangles, they code the valences of the processed vertices. If the variance of the valences is small in the mesh it can be compressed efficiently. From Eq. 1.8 we know that, vertices of large meshes have an average valence of six. Also by remeshing, irregular meshes can be converted into semi-regular or regular meshes whose valences are mostly six. So this method is very efficient on regular and semi-regular meshes.

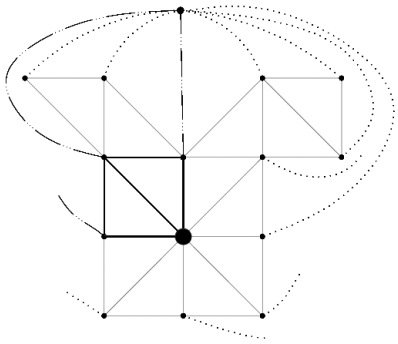
The TG coder first connects all the boundary vertices of the mesh with a dummy vertex as seen in Figure 1.7. Instead of selecting a starting triangle like EdgeBreaker, TG coder selects a starting vertex and outputs its valence with two other vertices of a triangle incident to the starting vertex. The triangle is marked as conquered and its edges are added to the cut-border. The concurrence of the incident triangles are iterated counter-clockwise around the focus vertex. When all the triangles incident to the focus vertex are conquered, the focus moves to the next vertex along the cut-border. This iterates until all the triangles are

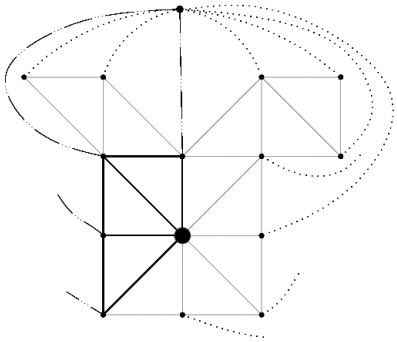
conquered. If dummy vertex becomes the focus vertex, the special symbol for dummy is output together with the valence of the dummy vertex.

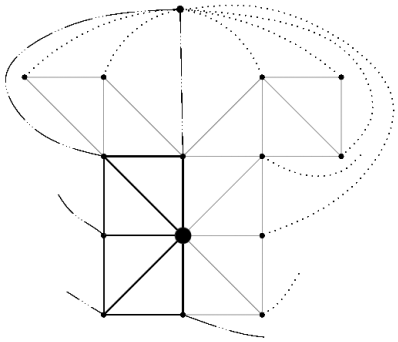


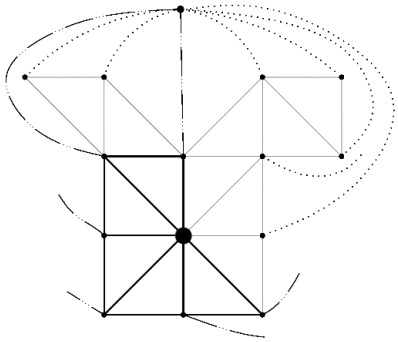












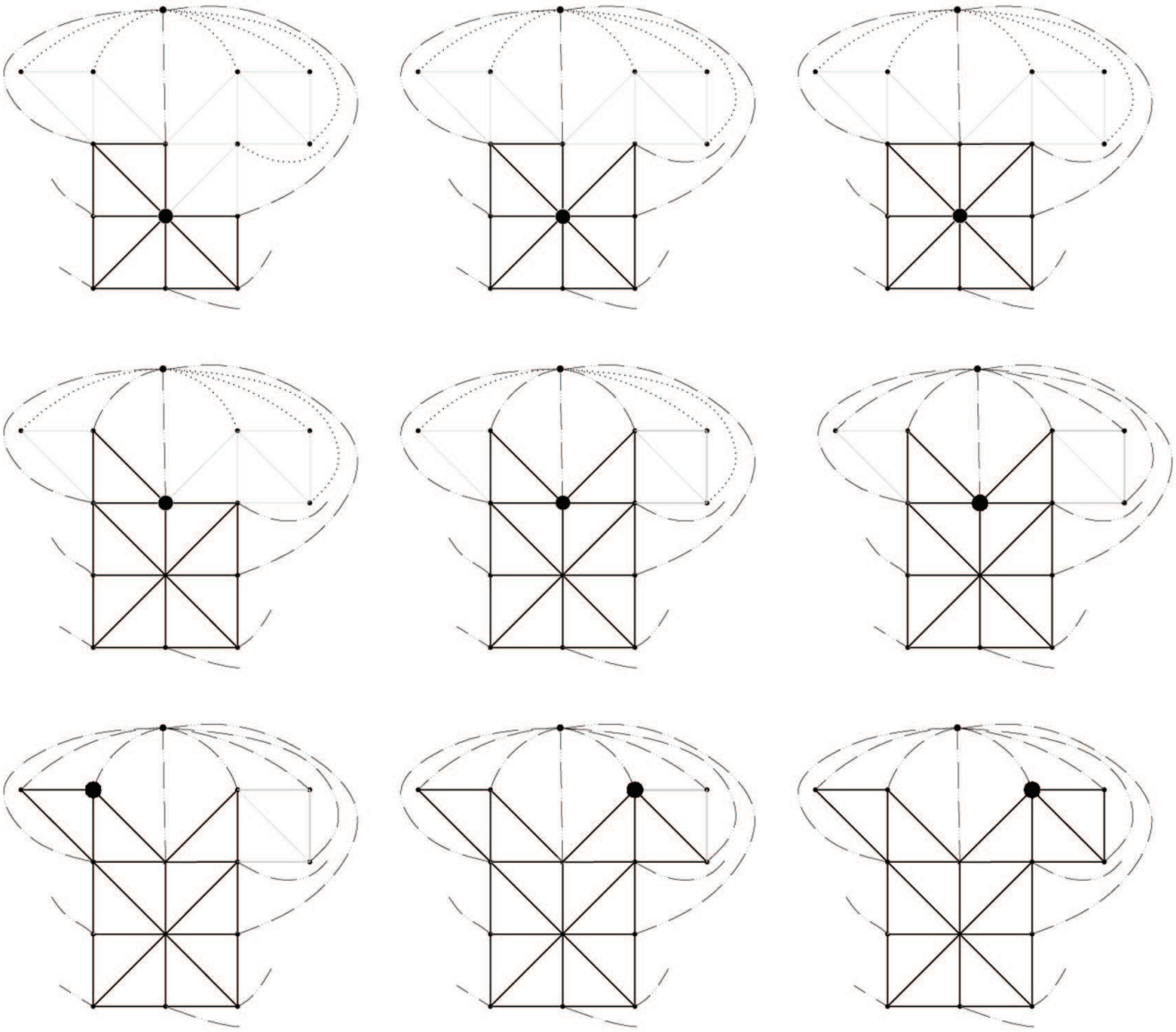


Figure 1.7: TG Encoding of the same mesh as Edgebreaker. The resulting codeword will be [8,6,6,4,4,4,4,4,6,4,5,Dummy 13,3,3].

A split situation can occur in this algorithm, too. It can arise when the cut-border is split into two parts. In this situation one of the cut-borders is pushed into the stack. The number of edges along the cut-border must also be encoded.

The split situation is an unwanted event since it complicates the encoding process. To reduce the number of the split situations, in [16] Alliez and Desbrun proposed an improved version of the TG coder. They approach the problem with the assumption of, "split situations are tend to arise in convex regions". So it is more reasonable to select focus vertices from concave regions instead of convex regions. While choosing the focus vertices, the algorithms pays attention to how many free edges does the vertex is neighboring. The vertex with minimum free edges is chosen. If there exists a tie between two vertices, number of the free edges of their neighbors will be taken into account.

Geometry Compression

The geometry data has a floating point representation. Due to the limitations of the human visual perception, this representation can be restricted to some precision which is called quantization. Nearly for all the mesh geometry codes, quantization is the first step of compression. The early works uniformly quantize each coordinate value separately in Cartesian plane. Also vector quantization of the mesh geometry is proposed in [31]. Karni and Gotsman also demonstrated the need for applying quantization on spectral coefficients [20] .

Sorkine et al. proposed a solution to the problem of minimizing the distortion in visual effects due to the quantization [32]. They quantize the vertices that are on the smooth parts of the mesh more aggressively. The basis behind this idea is the fact that human visual system is more sensitive to distortion in normals than geometric distortion. So, they try to preserve the normal variations over

the surfaces. In Figure 1.8 an illustrative comparison between quantization in Cartesian coordinates and delta-coordinates can be seen.

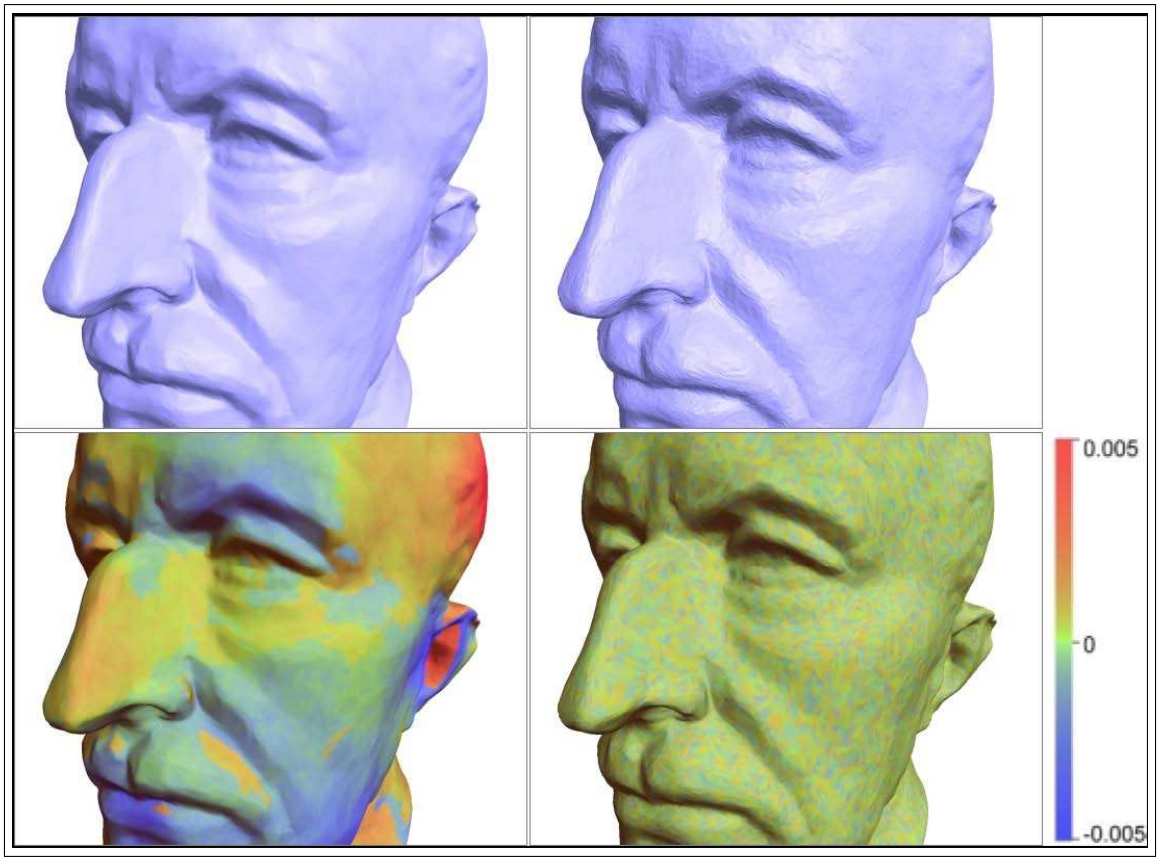


Figure 1.8: The delta-coordinates quantization to 5 bits/coordinate (left) introduces low-frequency errors to the geometry, whereas Cartesian coordinates quantization to 11 bits/coordinate (right) introduces noticeable high-frequency errors. The upper row shows the quantized model and the bottom row uses color to visualize corresponding quantization errors. (Reprinted from [32])

The prediction of the quantized vertices is a commonly used technique in mesh compression. To predict pixels from its neighbors is a well known technique in image processing. In some sense, the same approach is used in mesh compression. While traversing the mesh, position of the vertices are predicted from the formerly processed vertices. The prediction error, which is the difference between the predicted position and the real position, is coded. The prediction error tends to be a smaller value than the real position.

There exists several schemes to predict the vertex positions.The most known ones are; linear prediction using the weighted sum of the previous vertices (in

the order given by the connectivity coder) shown in Figure 1.9, parallelogram prediction shown in Figure 1.10, and multi-way prediction in [33].

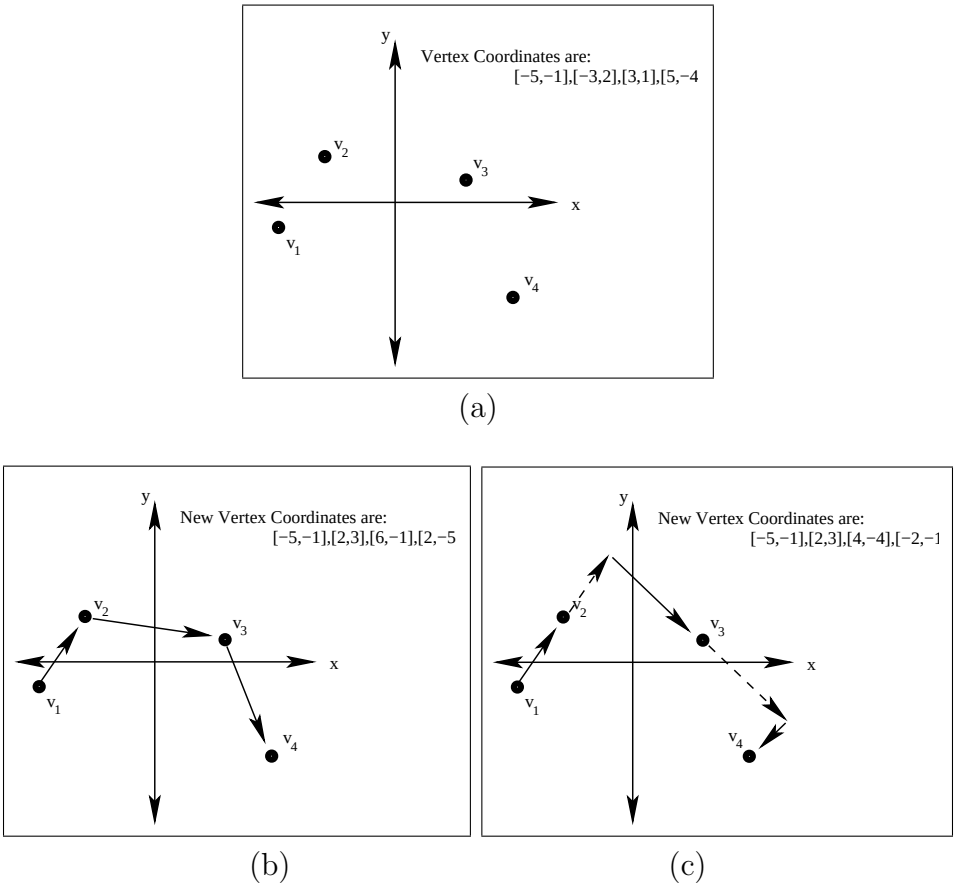


Figure 1.9: Linear prediction.

Touma and Gotsman introduced the *Parallelogram prediction* together with their TG coder [19]. This type of prediction is currently one of the most popular one. The basis of the idea is that, each edge has two incident triangles. So the position of a vertex $\mathbf{v_4}$ can be predicted using the vertices of the neighboring triangle, $\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}$ with respect to the opposing edge of $\mathbf{v_4}$. Vertices $\mathbf{v_2}, \mathbf{v_3}$ also composes an edge which is adjacent to both mentioned triangles. Figure 1.10 gives an illustration of parallelogram prediction algorithm. The vertex $\mathbf{v_4}$ is predicted using :

$$\hat{\mathbf{v}}_\mathbf{4} = \mathbf{v_2} + \mathbf{v_3} - \mathbf{v_1}, \tag{1.11}$$

and the error can simply be found by subtracting the predicted value from the actual position as:

$$\mathbf{e} = \hat{\mathbf{v}}_4 - \mathbf{v}_4. \tag{1.12}$$

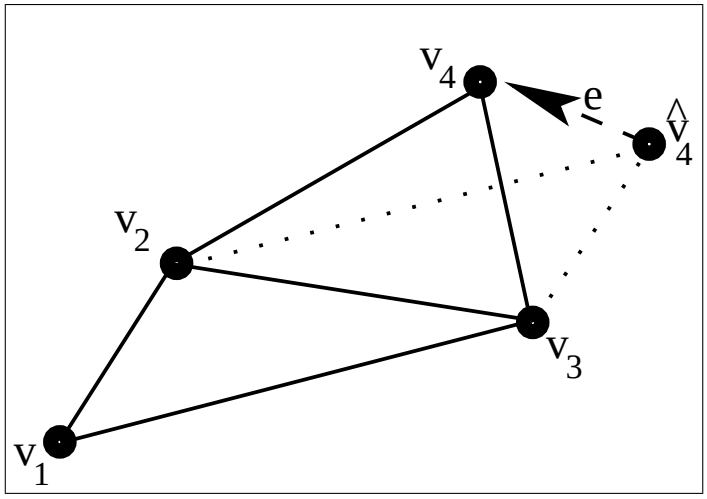


Figure 1.10: Parallelogram prediction.

In linear and parallelogram prediction schemes the vertex is predicted from one direction. This kind of prediction is not efficient for meshes with creases and corners. In [33] a prediction scheme, which uses the neighborhood information of the vertices to make predictions of the vertex positions, is proposed. In that scheme a vertex is predicted from all of its connected neighbors. The connected neighbors of a vertex can be found using the connectivity information of the mesh.

Another approach in geometry coding is, adapting the idea of spectral coding [20] to 3D signals so they can be used in compression of meshes. The idea is finding the best representatives for the mesh and code them. It is just like transforming an image into another domain and take the parts that has the most information. A known example of this kind of coding is DCT or wavelet coding

of the images. More attention is paid to the low frequency parts since they represent the image much better than the high frequency parts . Using the same convention, basis functions that represents the signal best are found.

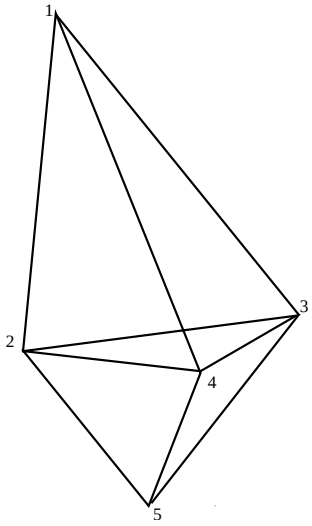
For geometry, the eigenvectors of the Laplacian matrix corresponds to the basis functions. The Laplacian **L** of a mesh is defined using diagonal valence matrix **VL** and adjacency matrix **A**. The Laplacian **L** can be calculated as :

$$\mathbf{L} = \mathbf{VL} - \mathbf{A}, \tag{1.13}$$

$$\mathbf{L} = \mathbf{UDU}^T, \tag{1.14}$$

$$\widetilde{\mathbf{V}} = \mathbf{U}^T\mathbf{V}. \tag{1.15}$$

Also Figure 1.11 shows **L**, **VL** and **A** for a simple mesh.



(a)

$$\begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

(b)

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

(c)

Figure 1.11: (a) A simple mesh matrix; (b) its valence matrix; and (c) its adjacency matrix.

Using Equation 1.14 eigenvectors of **L** can be evaluated. **U** contains the eigenvectors of **L** sorted with respect to their corresponding eigenvalues. The geometry is represented using a v *by* 3 matrix **V** where v is the number of the vertices of the mesh. **V** is projected into the new basis by Equation 1.15. The

rows of $\widetilde{\mathbf{V}}$ that are corresponding to low eigenvalues. Thus, they can be skipped and remaining coordinates can be encoded efficiently.

This representation also allows for progressive transmission. First important coordinates are coded first and send. Then rows corresponding to smaller eigenvalues are encoded and send. A low resolution mesh can be reconstructed from the first stream. Then the mesh can be refined as the lower priority rows arrive.

However there exists a disadvantage of using spectral coding. For larger matrices, decomposition in Equation 1.14 runs into numerical instabilities because many eigenvalues tend to have similar values. Also in terms of computation time calculating Equation 1.14 becomes an expensive job as the mesh grows.

Not only spectral coding but also the other mesh coding algorithms sometimes can not handle massive datasets like the model of *David of Michelangelo*. Those datasets should be partitioned into small enough meshes called *in-core*, in order to be compressed efficiently. In [35] Ho et al proposed an algorithm that partitions the massive datasets into small pieces and encode them using *EdgeBreaker* and *TG*. Also there is an overhead for stitching information exists. %25 increase with respect to the small mesh compression of the same tools. In [36] Isenburg and Gumbold made several improvements over [35] like avoiding to break the mesh,decoding the entire mesh in a single pass, streaming the entire mesh through main memory with a small memory foot-print. This is achieved by building new external memory data structures (*the out-of-core mesh*) dedicated for clusters of faces or active traversal fronts which can fit in this cores.

Besides cutting and stitching, some of the algorithms transform the meshes to other data structures and encode those new data structures. Formly mentioned *Spectral coding* and *Geometry images* [5] are examples to this type of algorithms. The basic idea in [5], is finding a parameterization of the 3D mesh to a 2D image and use image coding tools to code this image. To parameterize the mesh, it cut

along some of its edges and becomes topologically equivalent to a disk. This is the most challenging task of the algorithm. Finding those cuts is not a straightforward think to do.

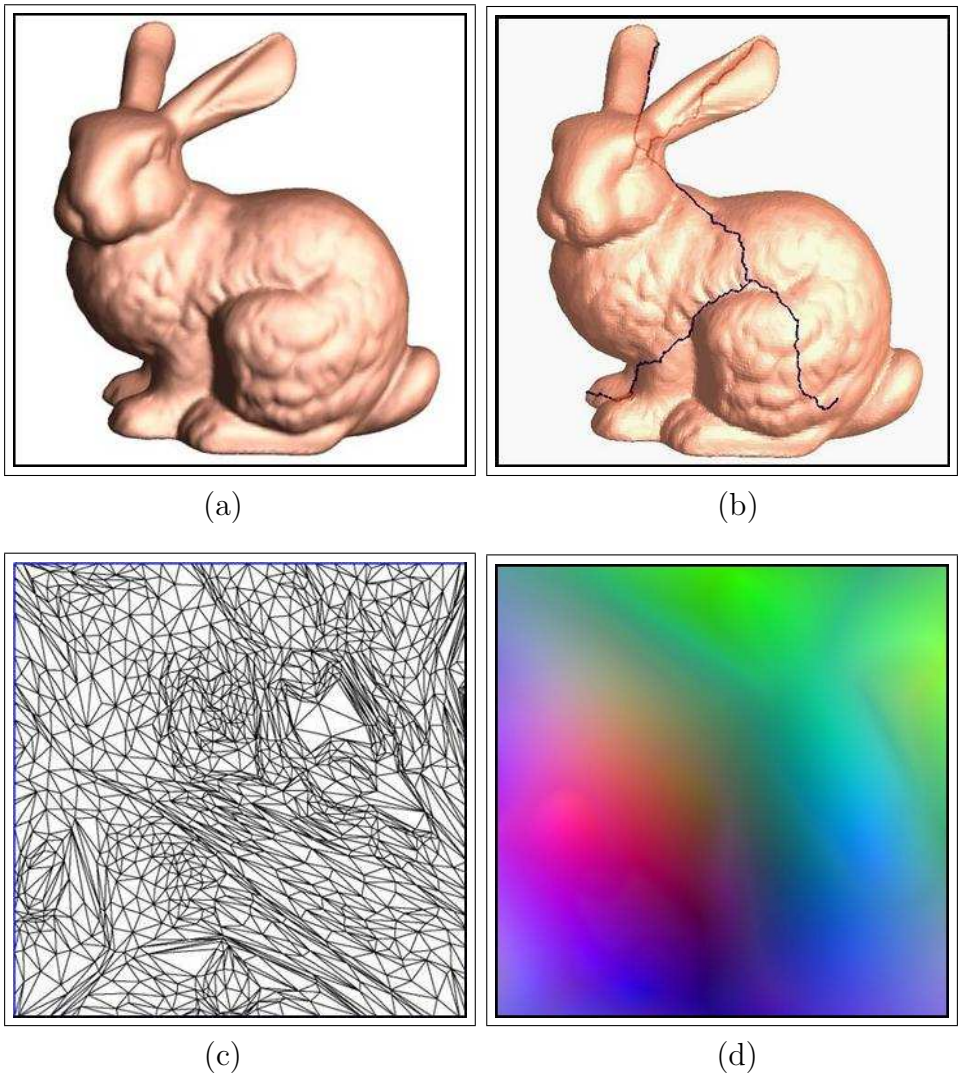


Figure 1.12: Geometry images: (a) The original model; (b) Cut on the mesh model; (c) Parameterized mesh model; (d) Geometry image of the original model. (Reprinted from [5]Data Courtesy Hugues Hoppe)

Parameterization domain is the unit square (2 *pixel by* 2 *pixel*). The pixel values of the parameterized mesh corresponds to points on the original mesh. Those points may either be vertex locations or surface points (points in triangular faces.). $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ coordinates of the points are written to the RGB channels of the image (Figure 1.12). Also a normal map image which defines all normals for the interior of a triangle, is stored. A geometry image is decoded simply by

drawing 2 triangles for each unit square and taking the RGB values as the 3D coordinates. Using the normal map the new mesh can be rendered. But the original mesh cannot be reconstructed. The decoded mesh will become like a remeshed version of the original one not exactly the original.

Remeshing is the process of approximating the original mesh using a more regular structure. There exists 3 types of meshes in this sense; Irregular, semi-regular, regular meshes. Most of the algorithms in the literature are adapting to the uniformity and regularity of the meshes [37]. So converting meshes into regular structures will bring the opportunity of more efficiently compressing the meshes. For example valence based compression approach in [16] codes regular meshes very efficiently since in a regular most the vertices has a valence of 6.

But remeshing is a hard task to accomplish. In [38], Szymczak introduced an idea of partitioning the mesh and resampling the partitioned surface. The resampled surface is retriangulated referencing the normals of the original mesh surface. Here both partitioning and retriangulation are computationally expensive and non-trivial tasks to do.

1.3.2 Progressive Compression

Lossless Compression Techniques

The basic idea behind Progressive Mesh (PM) compression is to simplfy a mesh and record the simplification steps. So the simplification can be inverted using the recorded information. PM coding is first introduced by Hoppe in [21]. He defined operation called *edge collapse and vertex split*. Figure 1.13 show the illustrates these operations.

An edge collapse operation merges two vertices incident to the chosen edge and connects all the edges connected to those vertices to the merged vertex. The

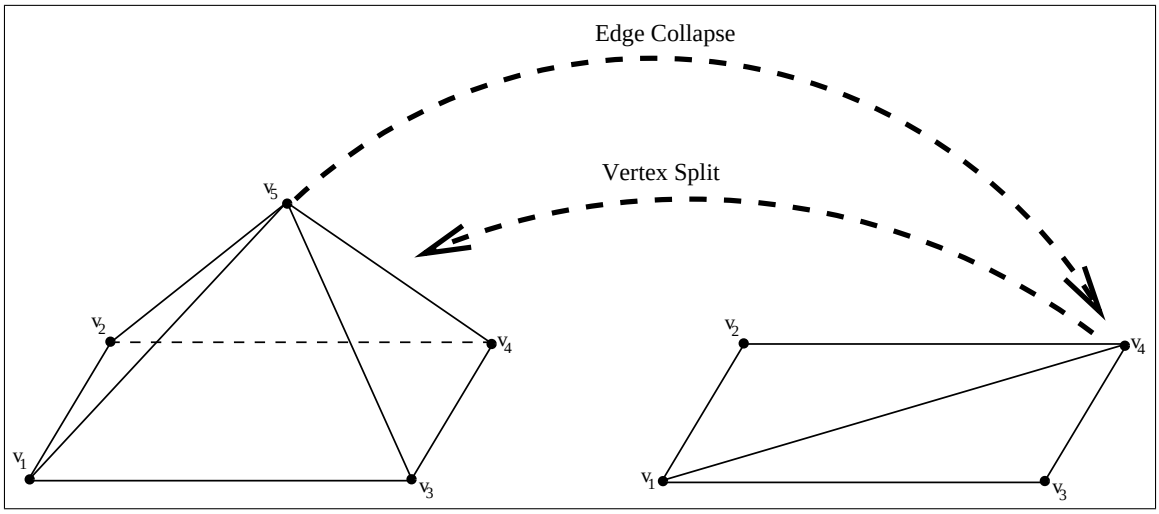


Figure 1.13: Edge collapse and vertex split operations are inverse of each other.

vertex split operation is the inverse of edge collapse. After a sequence of edge collapses, a simplified version of the original mesh is established (Figure 1.14(c)). One of the single rate coder can be used to encode this simplified mesh. The receiver side first decodes the simplified mesh and then uses the coming information to inverse the edge collapses by vertex split operation. Merged vertices are separated and those new vertices are connected to their neighbors

The selection of the edge to be collapsed next is an important issue since it affects the distortion of the simplified mesh. Hoppe[21] uses an energy function which takes into account the distortion that will be created by collapsing the edge. An energy value is assigned to each vertex. Incrementally an edge queue is created by sorting those energy values. The selection of the next edge to be collapsed is done according to this queue.

The *progressive simplicial complexes* approach in [39] extends the PM algorithm [21] to non-manifold meshes. Popovic and Hoppe [39] observed that using edge collapses resulting in non-manifold points, gives a lower approximation error for the coarse mesh. So they generalized the edge collapse method to vertex unification operation. Contrary to edge collapse, in vertex unification, the unified two vertices may not be connected by an edge. Arbitrary two vertices from

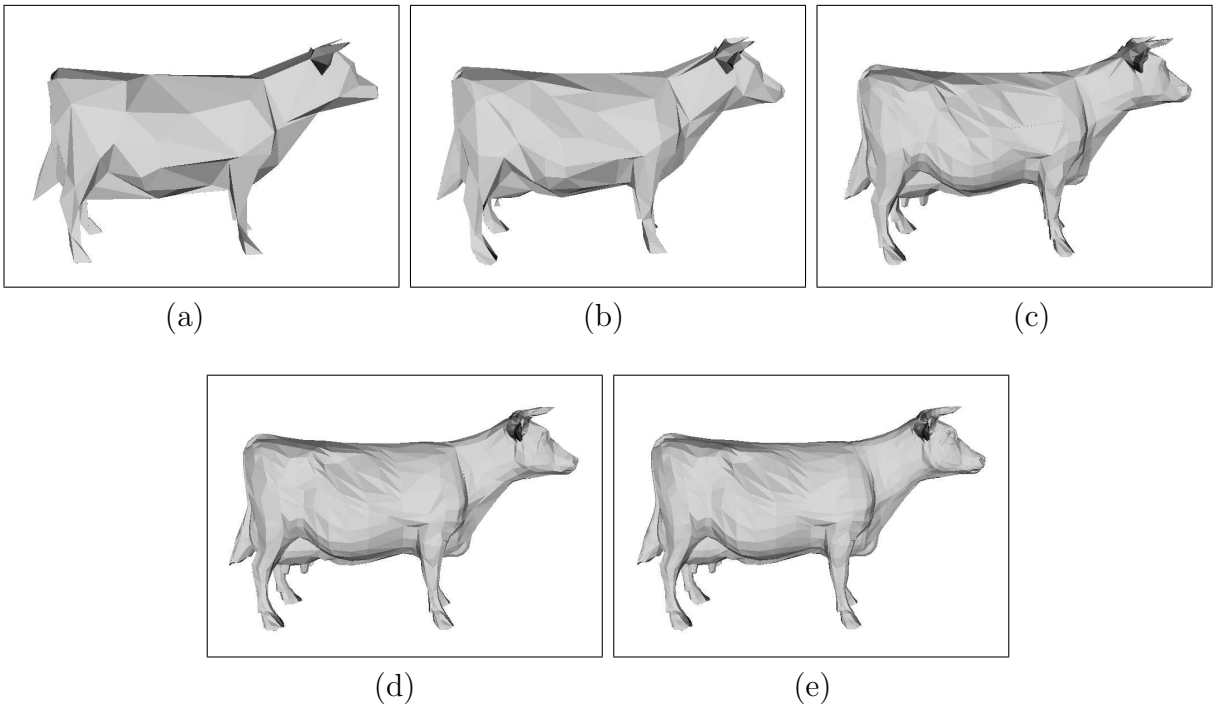


Figure 1.14: Progressive representation of the cow mesh model. The model reconstructed using (a) 296, (b) 586, (c) 1738, (d) 2924,and (e) 5804 vertices.

the mesh can be unified. Inverse operation is called generalized vertex split. The method can simplify arbitrary topology but has a higher bitrate than the PM algorithm since it has to record the connectivity of the region collapsed while unifying two vertices.

Another approach that is based on the PM method is introduced in [40] called *progressive forest split*(PFS). Contrary to the PM method, between to successive levels of detail there exists a group of vertex splits in PFS. Due to the split of multiple vertices in the mesh, cavities may occur in the mesh. Such cavities are filled with triangulation. The positions of newly triangulated vertices are corrected using translations. So each PFS operation encodes the forest structure, triangulation information and vertex position transitions. The PFS method is part of the MPEG-4 version 2 standard.

Compressed progressive meshes (CPM) method introduced by Pajarola and Rossignac is similar to the PFS method because it also refines the mesh using

multiple vertex splits called batches. But contrary to the PFS method in the CPM method cavities do not occur since the vertices who will result in new triangles after vertex split, are put in the batch. Connectivity compression is also easier in the CPM method, because of lack of cavities and as a result no triangulation information is needed. Transition information in the PFS method is replaced with prediction information. Position of the new vertices are predicted from their neighbors using butterfly scheme.

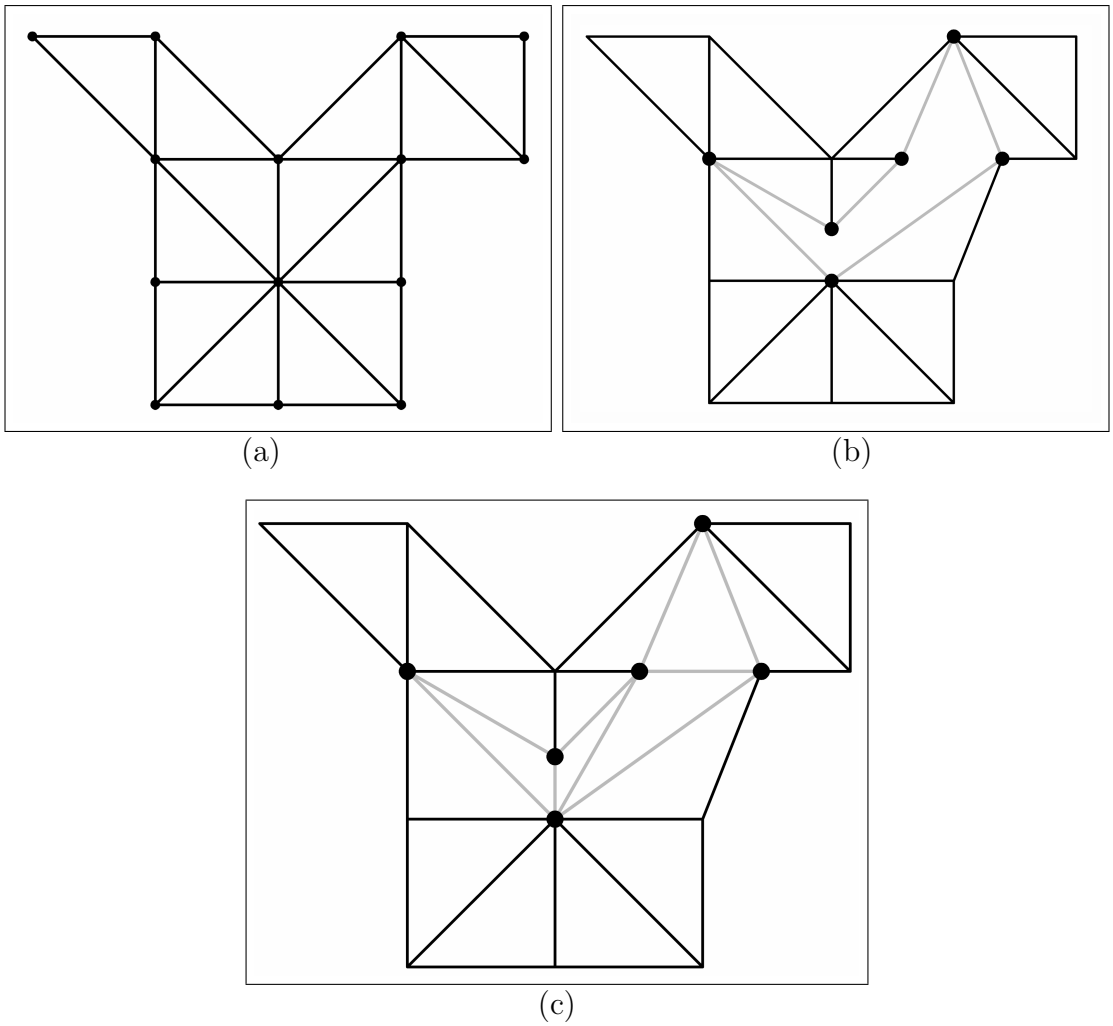


Figure 1.15: Progressive forest split. (a) initial mesh; (b) group of splits; (c) remeshed region and the final mesh.

Techniques mentioned so far are all the PM method based approaches. Alliez and Desbrun proposed in [41] a method which uses the valence information of the vertices of the mesh. From Equation 1.8 it is known that average valence

value of a mesh is 6. They observe that the entropy of the mesh is closely related with the distribution of this valence. Their proposed algorithms has two parts: *decimation conquest* and *cleaning conquest*.

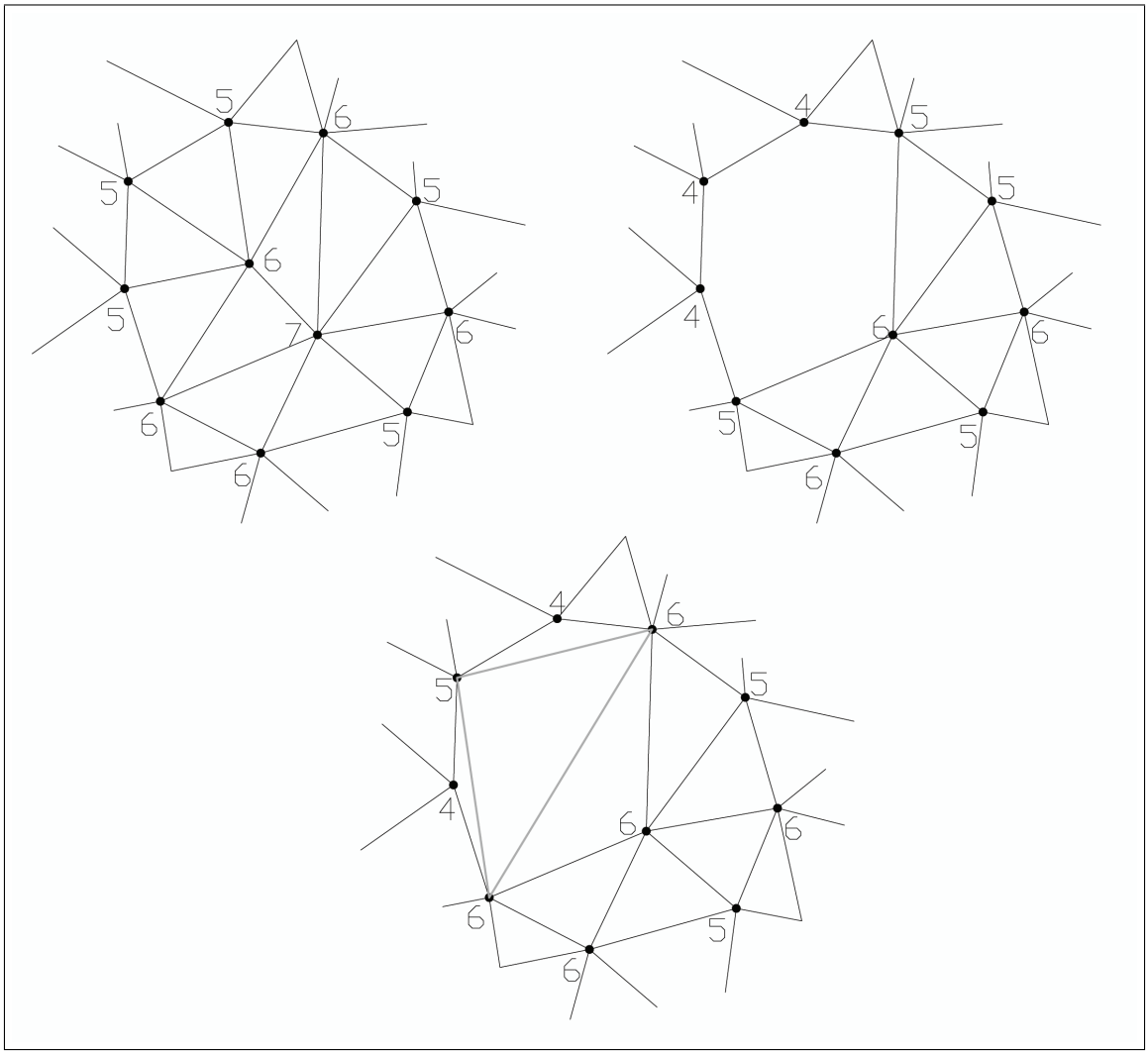


Figure 1.16: Valence based progressive compression approach.

The decimation conquest first subdivides the mesh into patches. Each patch consists of triangles which are incident to a vertex as shown in Figure 1.16 (a). Then the encoder enters the patches one-by-one, removes the common vertex and re-triangulates the patch as shown in Figure 1.16(b-c). The valence of the removed vertex is output. This is applied to all the patches in the mesh. Then the cleaning conquest decimates the remaining vertices with valence 3. This

algorithm preserves the average value around 6. The mesh geometry is encoded using barycentric prediction, and the prediction error is coded.

Lossy Compression Techniques

The basic idea behind lossy mesh compression is existence of another surface which is a good approximation of the original one and more capable for compression. In lossy compression the original geometry and the connectivity information is lost. The distortion between the original and its representative models is measured as the geometric distance between surfaces of those.

Multiresolution analysis and wavelets are the key methods in lossy mesh compression algorithms. They allow to decompose a complex surface into coarse representations together with refinements stored in the wavelet coefficients. Surface approximations at different distortions levels can be obtained by discarding or quantizing wavelet coefficient at some level. Since the mesh is decomposed into more energetic low frequency (multiresolution mesh levels) and low variance high frequency (wavelet coefficients) parts, an more compact compression can be achieved.

Wavelets are known as signal processing tools on cartesian grids like, audio, video, and image signals. Lounsberry et al. in [2] extended the wavelets to 2-manifold surfaces of arbitrary type. Due to their non-regularity, it is impossible to adapt wavelet transform to irregular meshes [42]. However, by creating semi-regular meshes using remeshing process, the mesh can be made regular. On this regular structure an extension of the wavelet transform can be used.

Here the remeshing process is implemented by subdivision. The algorithm starts with the coarse representation of the original model called base mesh. Each triangle of the base mesh subdivided into 4 triangles by creating vertices on the edges of the face. The positions of those vertices represent again surface

samples of the original model. So the subdivided coarse mesh becomes more and more similar to the original model. However, now the new representation is a semi-regular mesh whose connectivity is regular.

After obtaining the semi-regular structure, the model is now ready for progressive compression. The idea in multiresolution mesh analysis is to use wavelet transform between two resolution levels to code the prediction errors in the vertex locations. While going from the fine to course resolution the positions of the deleted vertices are predicted using subdivision. The difference ($\mathbf{v_e}$) between the predicted($\mathbf{v_p}$) and the decimated vertex position ($\mathbf{v}$) is stored as wavelet coefficients as follows :

$$\mathbf{v_e} = \mathbf{v_p} - \mathbf{v}, \tag{1.16}$$

where $\mathbf{v}, \mathbf{v_e}, \mathbf{v_p} \in R^3$.

The process is similar to the high-pass filtering since the wavelet coefficients store the details of the mesh vertex positions. In the coarse to fine transition, first the vertex positions are predicted using subdivision and then repositioned using the wavelet coefficients. Discarding some of the wavelets to achieve better compression rates is possible. However, it is obvious that this brings distortion to the reconstructed model.

The third important part in progressive compression of the meshes is the coding of the base mesh and wavelet coefficients. As mentioned base mesh coding can be done using one of the single rate mesh coder. Wavelet coefficients are coded using entropy coders. Zero-tree coders would efficiently code this type of data since the coefficient tend to decrease from coarse to fine levels.

Khodakovsky in [43] proposed a lossy progressive mesh compression approach that uses MAPS [45] algorithm to remesh the original model a semi-regular mesh.

In [43] Loop scheme [34] is used to predict the vertex positions. In this algorithm the wavelet coefficients are 3D vectors as seen in Equation 1.16.

In [44] Khodokovsky and Guskov used normal meshes [23] and NMC wavelet coder to encode meshes. NMC coder uses normal meshes to encode the wavelet coefficients. Wavelet coefficients are shown as scalar offsets (normals) in the perpendicular direction relative to the face. So wavelet coefficients become scalar values instead of 3D vectors. They used unlifted butterfly scheme [47] as predictor as it is also used while normal remeshing.

Marón and Garcia proposed a method that generates the base mesh using Garland's quadratic-mesh simplification [48]. They make predictions using the butterfly scheme in [47]. The prediction errors are coded as wavelet coefficients. Since the normal component of the wavelet coefficients carry more information than the tangential components, wavelet coefficients are finely quantized in normal direction and coarsely quantized in tangential direction. A bit-wise interleaving of three detail components gives a scalar value so that a stream of scalar values is generated. This stream is coded using the SPIHT algorithm [49].

1.4 Contributions

This thesis proposes a framework to compress 3D models using image compression based methods. In this thesis two methods are presented:

- a projection method to transform meshes into images objects,

- a connectivity based adaptive wavelet transformation (WT) scheme that can be embedded into a known image coder and this WT scheme is used in the coding of image objects.

The proposed projection method is computationally easier than the cutting and parameterizing the mesh proposed in [5]. The proposed connectivity guided adaptive wavelet transform newly defines the pixel neighborhoods. Thus, it is more efficient than using ordinary wavelet transform schemes on the projection images

1.5 Outline of the Thesis

In the second chapter of this thesis, the proposed algorithm is explained in detail. The components of the algorithm include the like projection operation, connectivity guided adaptive wavelet transform, Set Partitioning in Hierarchical Trees (SPIHT), JPEG2000, map coding and reconstruction are explained. The main idea is to find a way to use image coders for mesh coding and embed adaptiveness to image coder so that better bit rates are achieved. In this thesis, this aim is achieved by using adaptive versions of the SPIHT and JPEG2000 image coders.

In the third chapter, mesh coding results of our algorithm and comparison with the algorithms in literature will be presented. Chapter 4 involves the conclusions related to this thesis.

Chapter 2

Mesh Compression based on Connectivity-Guided Adaptive Wavelet Transform

This chapter is composed of several sections dealing with different parts of the mesh coding algorithm. These sections include mesh to image projection, connectivity-guided adaptive wavelet transform, etc. The combination of all of those parts results in a complete mesh coder which can encode 3D models using any wavelet based image coders with minor modifications.

The proposed mesh coding algorithm introduces an easy way of converting meshes to image like representations so that 2D signal processing methods become directly applicable to a given mesh. After the projection, the mesh data becomes similar to an image whose pixel values are related to the respective coordinates of the vertex point to the projection plane.

The mesh data on a regular grid is transformed into wavelet domain using an adaptive wavelet transform. The idea of adaptive wavelets [50] is a well known and proved to be a successful tool in image coding. Exploiting the directional

neighborhood information between pixels, adaptive wavelet transform beats its non-adaptive counterparts. Thus, instead of using non-adaptive wavelet transform we come up with the idea of defining an adaptive scheme so that neighborhood information of the vertices can be better exploited.

2.1 3D Mesh Representation and Projection onto 2D

A mesh can be considered as an irregularly sampled discrete signal. This means that the signal is only defined at vertex positions. In the proposed approach, the mesh is converted into a regularly sampled signal by putting a grid structure in space; corresponding to resampling the space with a known sampling matrix and quantization.

2.1.1 The 3D Mesh Representation

The 3D mesh data is formed by geometry and connectivity information. The geometry information of the mesh is constituted by vertices in $\mathbf{R}^3$. The 3D-space, where the geometry information of the original mesh model is defined, is $\mathbf{X}' = (x',\ y',\ z')^T \in R^3$. The vertices of the original 3D model is represented by $\mathbf{v}'_i = (x'_i, y'_i, z'_i)^T,\ i = 1,\ \ldots,\ v$ where v is the number of the vertices in the mesh.

First, the space that we use is normalized in $\mathbf{R}^3[-0.5,\ 0.5]$ by,

$$\mathbf{X} = (x,\ y,\ z)^T = \alpha\mathbf{X}',\ \alpha = (\alpha_x,\ \alpha_y,\ \alpha_z) \in (0, 1). \tag{2.1}$$

This results to a normalization in the coordinates of the mesh vertices. The normalized mesh vertices are represented as,

$$\mathbf{v}_i = (x_i,\ y_i,\ z_i)^T = (\alpha_x x'_i,\ \alpha_y y'_i,\ \alpha_z z'_i),\ i = 1,\ \ldots,\ v. \tag{2.2}$$

Connectivity information is represented as triplets of vertex indices. Each triplet corresponds to a face of the mesh. Faces of the mesh can be represented as,

$$\mathbf{F}_i = (\mathbf{v}_a,\ \mathbf{v}_b,\ \mathbf{v}_c)^T,\ i = 1,\ \ldots,\ f\ \ \&\ \ a, b, c\ \in \{1,\ \ldots,\ v\}, a \neq b \neq c \tag{2.3}$$

where f is the number of the faces and v is the number of the mesh. So a mesh in $\mathbf{R}^3$ can represented as,

$$\mathbf{M}\ =\ (\mathbf{V}, \mathbf{F})\ , \tag{2.4}$$

where V is the set of vertices of the mesh and F is the set of faces of the mesh.

2.1.2 Projection and Image-like Mesh Representation

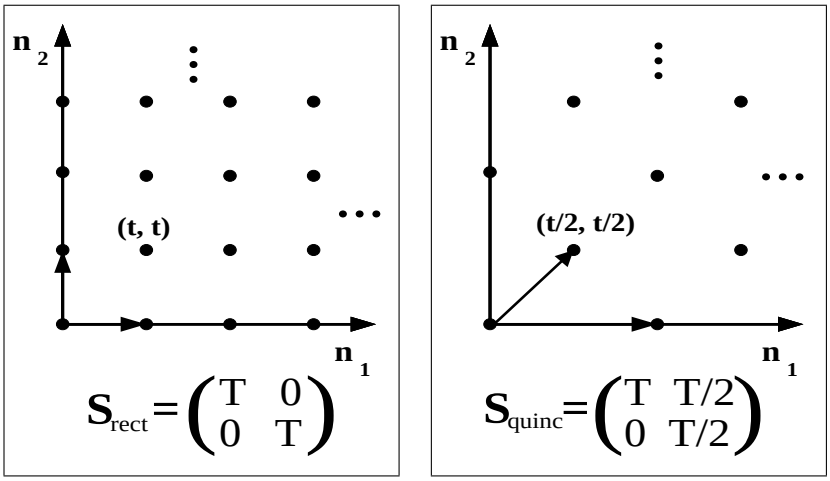
The 3D mesh representation is transformed into a 2D image-like representation by projecting the mesh data onto a plane. In this thesis a projection is defined as the transformation of points and lines in one plane onto another plane by connecting corresponding points on the two planes with parallel lines [53]. This is called *Orthographic projection*. The points of projection in our algorithm are vertices of the mesh. The selected projection plane is defined as $\mathbf{P}(u, w)$. The projection plane $\mathbf{P}$ is discritized using the sampling matrix $\mathbf{S}$. Different sampling matrices $\mathbf{S}$, which are seen in Figure 2.1, can be defined as:

$$\mathbf{S}_{rect} = \begin{pmatrix} T & 0 \\ 0 & T \end{pmatrix}, \tag{2.5}$$

for rectangular sampling and

$$\mathbf{S}_{quinc} = \begin{pmatrix} T & T/2 \\ 0 & T/2 \end{pmatrix}, \tag{2.6}$$

for quincunx sampling.



(a) (b)

Figure 2.1: (a) 2D rectangular sampling lattice and 2D rectangular sampling matrix (b) 2D quincunx sampling lattice and 2D quincunx sampling matrix.

Due to its shape quincunx sampling lattice makes better approximations of the projected mesh vertices. But implementation of the rectangular is more straight-forward and computationally easy. After projection operation, the grid points sampled using quincunx sampling lattice must be transformed to a rectangular arrangement since the pixels of the image-like representation lined up in a rectangular manner. The projection of a 3D mesh structure mainly depends on two parameters of the vertex; coordinates of the vertex and the perpendicular distance of the vertex to the selected projection plane.

Let $\check{\mathbf{v}}_\mathbf{i}$, which is a 2D vector, be the projection of $\mathbf{v_i}$ onto the plane $\mathbf{P}$. Furthermore let $\mathbf{d}_i$ be the perpendicular distance of $\mathbf{v_i}$ to the plane $\mathbf{P}$. The illustration of the projection operation can be seen in Figure 2.2. A simple 3D mesh of a rectangular prism with its vertices on its corners is projected onto a plane.

In the *Projection Image*, the pixel positions are determined using the respective positions of the vertices to the projection plane. The values of the pixels are determined using the perpendicular distance of the vertices to the projection plane. Thus, using different projection planes creates different image-like representations as seen in Figure 2.3.

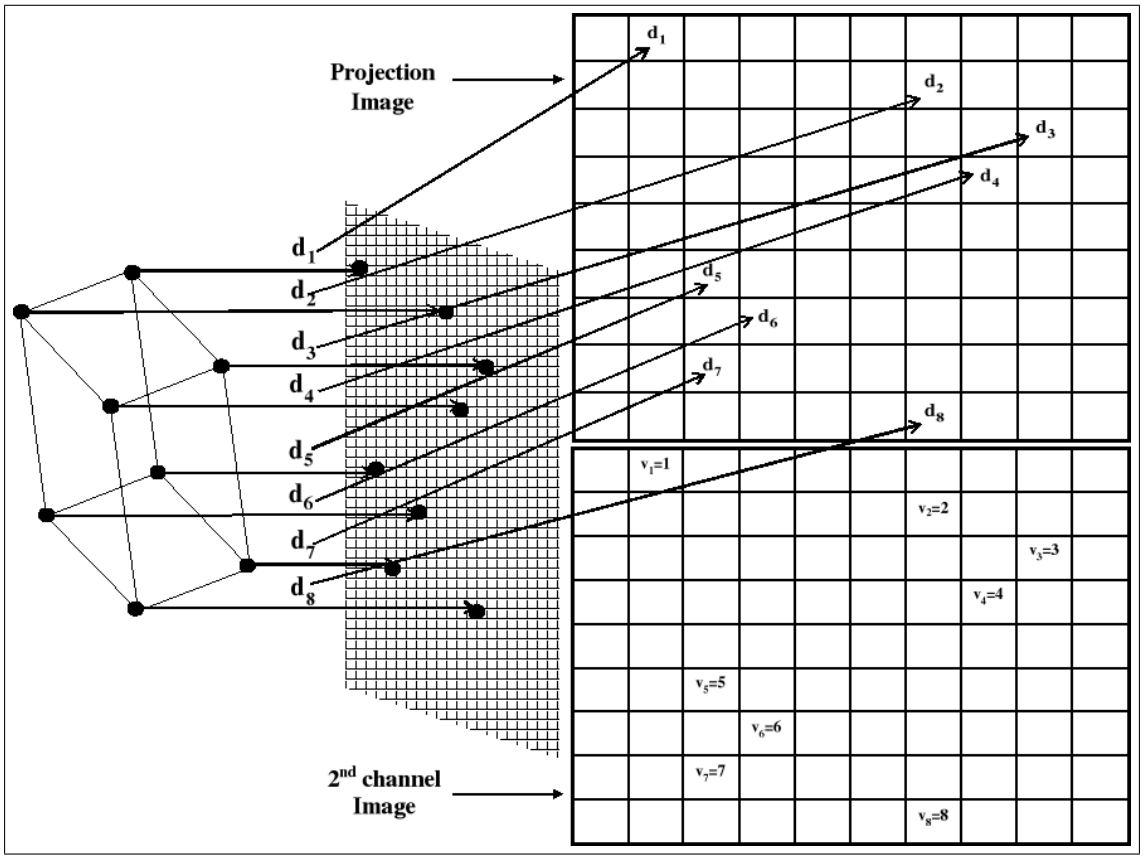
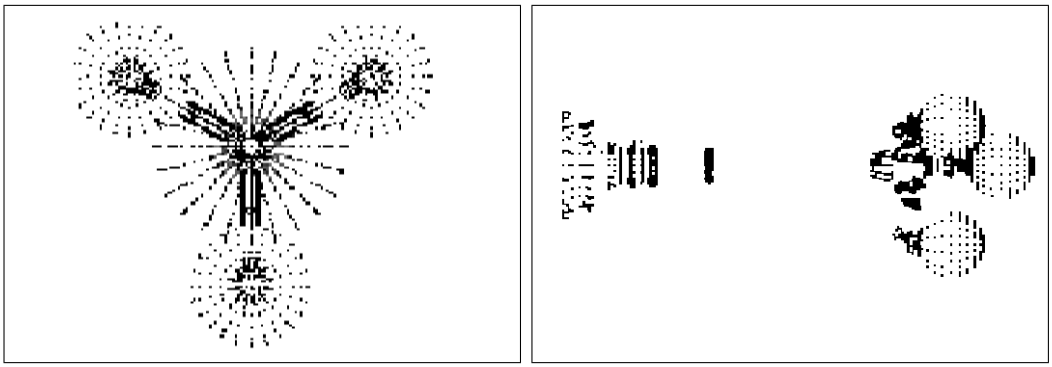


Figure 2.2: The illustration of the projection operation and the resulting image-like representation.

The determination of vertex-grid point correspondence is the most crucial task in the projection operation. The vertices that can be assigned to a grid point $\mathbf{n} = [n_1, n_2]$ forms a set of indices J defined by:

$$\mathbf{J} = \left\{ i \;\middle|\; |\check{\mathbf{v}}_i - \mathbf{S}\,\mathbf{n}^{\mathbf{T}}| < T/2 \;\forall\; n_1, n_2 \right\} \tag{2.7}$$



(a) (b)

Figure 2.3: Projected vertex positions of the mesh; (a) projected on XY plane; (b) projected on XZ plane.

where $\mathbf{S}$ is a sampling matrix and $\mathbf{n} = [n_1, n_2]$ represent the indices of the discrete image as shown in Figure 2.2. Sampling matrix $\mathbf{S}$ determines the distance between neighboring grid points, and can be defined as a quincunx or a rectangular sampling matrix [1].

Then the 3D mesh is transformed to two 2D image-like representations. The first image stores the perpendicular distances of the vertices to the respective grid points on the selected planes as :

$$\mathbf{I}_1[n_1, n_2] = \begin{cases} \mathbf{d}_i, & i \in J, \\ 0, & otherwise. \end{cases} \tag{2.8}$$

The second image holds the indices of the vertices as follows :

$$\mathbf{I}_2[n_1, n_2] = \begin{cases} i, & \mathbf{d}_i = \mathbf{I}_1[n_1, n_2], \\ \\ 0, & otherwise. \end{cases} \tag{2.9}$$

The first channel image is then wavelet transformed using our newly defined *Connectivity-Guided Adaptive Wavelet Transform*. The transformed image is then encoded using SPIHT [49] or JPEG2000 [52]. The second channel image is converted to a list of indices. This list is differentially coded and sent to the other side.

Using Equations. 2.8 and 2.9 we try to find a pixel-vertex correspondence for each vertex. However as seen from Equation 2.7, sometimes more than one vertex have the same projection in the image-like representation. Thus, one of the vertices is chosen for the calculation of the pixel value and the others are discarded. By increasing the number of projection planes or decreasing the sampling period by changing the sampling matrix **S** we can handle more vertices.

Both methods increase the reconstruction quality since they decrease the number of lost vertices. However, they will lead to a decrease in the compression ratio. In our approach we used one densely sampled plane. The recovery of the lost vertices is handled by connectivity - based interpolation.

2.2 Wavelet Based Image Coders

Multiresolution is a very important concept in image processing. The main idea of multiresolution methods is to analyze an image in different resolutions. Wavelet transform is an efficient tool of multiresolution signal analysis. Using a filterbank containing a high pass and a low pass filter, an approximation and a residual part of an image is calculated, respectively. The approximation part of the image has nearly the same pixel value distribution with the original image. But the values of the pixels in residual part becomes concentrated around a mean value with a small variance. This kind of signal is more suitable for encoding since most of the information is now concentrated in a smaller pixel value region. As the wavelet transform goes on new high pass parts are created thus, the pixel value distribution changes and signal becomes more suitable for encoding.

Choice of the wavelet basis, number of decomposition levels and design of the quantizer are very important issues in wavelet based image coders. Using complex wavelet functions would slow down the process but the resultant coefficients are less correlated.

Increasing the number of multiresolution decomposition level slows down the process since more wavelet transform stages have to be computed during the compression. But it leads to better compression results since as the decomposition level increases the number of the coefficients that can be coarsely quantized increases, too.

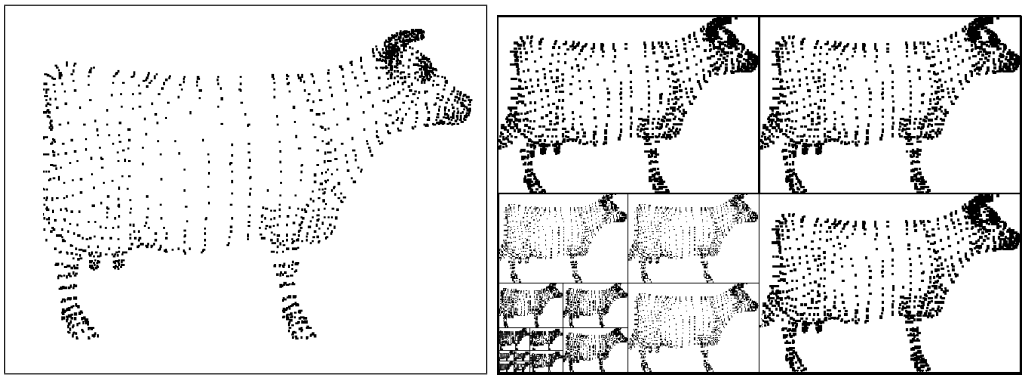
Quantizer design is the most critical issue that affects the coding compression and reconstruction error issues. Instead of using uniform quantizers, the nature of the signal must be examined carefully, and a non-uniform quantizer which takes care of the important coefficients should be used. Also adapting the size of the quantization intervals from scale to scale results in a better rate-distortion.

In this thesis among several wavelet-based mesh coders two well known ones, namely SPIHT and JPEG2000 are used. Zero-tree coding approach is very suitable for the compression of the proposed image-like representation. SPIHT is chooses because it is one of the most successful algorithm that uses zero-tree coding. JPEG2000 is chosen because it is a new technology image coder and seems to give very good results on images.

2.2.1 SPIHT

In Discrete Wavelet Transform (DWT), a 1-D discrete signal is processed by a filterbank having a complementary half-band low-pass and high-pass filter pair. Outputs of the two filters are downsampled and two subsignals are obtained. The high-band subsignal contains the first level wavelet coefficients of the original signal corresponding to the normalized frequency range of $[\pi/2, \pi]$. The low-band subsignal is processed using the same filterbank once again; the second level wavelet coefficients covering the frequency range of $[\pi/4, \pi/2]$ and the low-low band subsignal covering $[0, \pi/4]$ are obtained. In a typical application, this process is repeated several times. Extension of the 1-D DWT to the 2-D signals

can be carried out in a separable and non-separable manner. In the separable case, the 2-D signal is first processed horizontally row by row by the 1-D filterbank and two subimages are obtained. These two subimages are then filtered column by column by the same filterbank and, as a result, four subimages, low-low, high-low, low-high and high-high subimages are obtained (the order of filtering is immaterial). High-band subimages contain the first-level wavelet coefficients of the original 2-D signal. The low-low subimage can be processed by the 2-D filterbank recursively.



(a) (b)

Figure 2.4: (a) Original image (b) 4-level wavelet transformed image.

A 1-D DWT can be extended into a 2-D case in a non-separable manner and 2-D signals in quincunx grids can be analyzed using the fast wavelet transform [54]. Figure 2.4 shows an image (a) and its bands of the 4-level wavelet transform (b). The Embedded Zero-Tree Wavelet Coding (EZW) takes advantage of the zeros in multi-scale wavelet coefficients or a 2-D signal [51]. Wavelet coefficients corresponding to smooth portions of a given signal are either zero or close to zero. Wavelet coefficients only differ from zero around the edges of a given image. Based on this assumption, zeroing some of the small-valued wavelet coefficients would not cause much distortion while reconstructing the image. EZW also takes advantage of the relation between multiresolution wavelet coefficients obtained using the 2-D wavelet tree.

A typical natural image is low-pass in nature. As a result most of the wavelet coefficients are zero or close to zero except the coefficients corresponding to edges and textures of the image. On the other hand the data in image-like signals that we extract from meshes are sparse. Most of the grid points have zero values as there are very few vertices around smooth parts of the mesh. Filtering the mesh using a regular low-pass or high-pass filter of a wavelet transform spreads an isolated mesh value in the wavelet subimages. One can use the lazy filterbank of the lifting wavelet transform shown in Figure 2.5 in which the filter impulse responses are trivial, $h_l[n] = \delta[n]$ and $h_h[n] = \delta[n-1]$. Therefore, using the lazy filterbank has advantages over the regular filterbanks in mesh images. Using the lazy filterbank rearranges the mesh image data into a form that EZW or SPIHT can exploit. As in the case of a natural image, most of the high-band wavelet coefficients turn out to be zero and there is a high correlation between the high-band subimages.

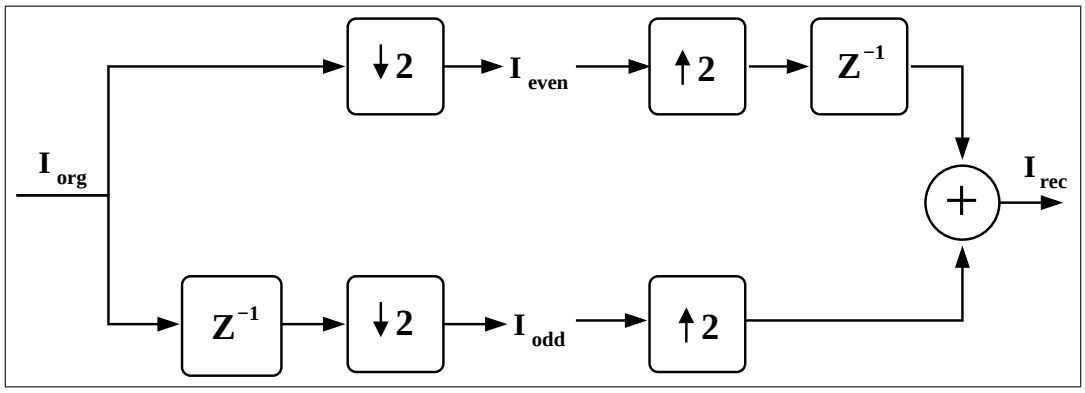


Figure 2.5: Lazy filter bank.

EZW, which is the predecessor of SPIHT, is also an algorithm based on wavelet transform. An embedded code is a list of binary decisions. The original signal is obtained from an initial signal according to the list of decisions. The crucial aspect of embedded coding is that the decisions are ordered according to their importance. This makes EZW a coder that produces progressive compression-like coding.

In EZW the bitstream can be truncated anywhere according to the users' needs. The longer the parts of the bitstream that are chosen, the closer the reconstructed initial signal gets to the original signal.

A dependency exists between the wavelet coefficients, as shown in Figure 2.6 (a). This is the root-node hierarchy. The roots correspond to lower bands and the nodes correspond to higher bands. In most cases, the coefficients in the lower-level nodes are smaller than the coefficients in their roots. Figure 2.6 (b) shows the tree structure of the same hierarchy.

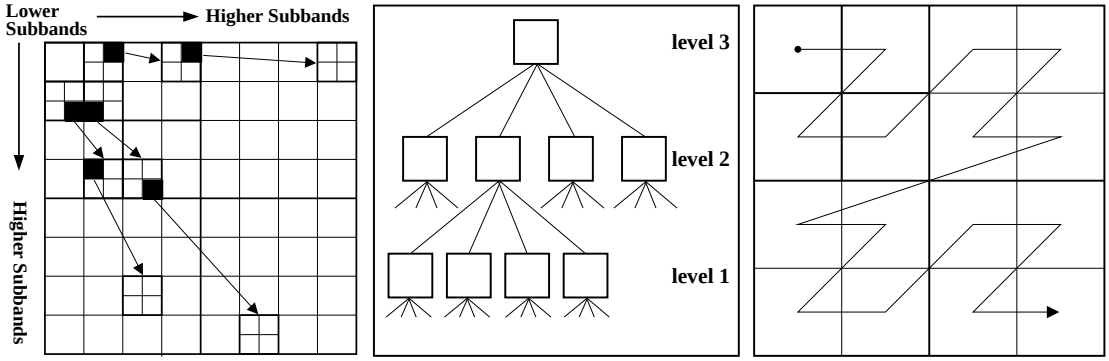


Figure 2.6: (a) The relation between wavelet coefficients in EZW; (b) the tree structure of EZW; (c) the scan structure of wavelet coefficients in EZW.

When a coefficient in the root is smaller than a threshold, its nodes contain even smaller values. Thus, this branch of the tree, which is called a *zero-tree*, can be pruned and coded easily. EZW checks for zerot-rees in the transformed image and codes them with a special symbol.

In principle, SPIHT carries out the same procedure. It takes the wavelet transform of the image until a user-defined scale is reached. It creates three lists by scanning the wavelet domain image in the order shown in Figure 2.6 (c). These lists are List of Significant Pixels (**LSP**), List of Insignificant Pixels (**LIP**), and List of Insignificant Sets (**LIS**). First, the coefficients in LIP are compared to a chosen threshold. If the value of the coefficient is bigger than the threshold

it is put into LSP. The same procedure is carried out for the coefficients in LIS. This is called the *sorting pass*.

In the *refinement pass*, the n^{th} most significant bit of each entry in LSP is transmitted except those included in the last sorting pass [55]. The details of EZW and SPIHT can be found in [51, 56, 49].

2.2.2 JPEG2000 Image Coding Standard

Unlike its predecessor JPEG, JPEG2000 is not a DCT based compression approach. By using wavelets, JPEG2000 provides increased flexibility in both the compression of the images and access to the compressed data. Any portion of the image can be extracted without a need of extracting the whole compressed data. Quantization of the transform coefficients are adapted to the scale and subband of the transformed image. At last arithmetic coding takes place to code the quantized coefficients.

The compression process starts with DC level shifting of the samples so that the mean of the image becomes zero. Then the color channels are decorrelated using *RGB* to *YCbCr* color transformation. After those the image is partitioned into tiles. Tiles are rectangular parts of the image which are coded individually. This approach gives the algorithm the opportunity of extracting individual parts of the image independent from the other parts.

Each tile is transformed into subbands using *Discrete Wavelet Transform*(DWT). Either *5-3 biorthogonal* or *9-7 biorthogonal* [57],[58] wavelet-scaling basis are used for DWT. The transform is computed either by fast wavelet transform or lifting structure. Number of the transformation coefficients in a tile is equal to the number of the pixels in that tile. But after the transformation most of the information is concentrated in few coefficients in most natural images. Using a quantizer those coefficients that have little information can be

suppressed (big quantization steps) and the ones with the more information content stay untouched or disturbed as least as possible (small quantization steps). The quantization process is implemented on each individual tile separately.

The final step is creating a bitstream from the quantized coefficients. First DWT transformed subbands are partitioned into *code-blocks*. Each code block is encoded bit-plane at a time. Bit planes are then coded starting from the most significant bit-plane. *Layers* are created from those coded bit-lanes and then the layers are partitioned into packets.

The decoder of JPEG2000 simply reverses the operations done by the encoder. The number of the encoded and decoded bit-plane levels may change, according to the user's requirements. Any nondecoded bits are then set to zero. This gives a progressive nature to the algorithm. The image can be reconstructed without having all the bit-planes.

After inverting the quantization operation the tiles are transformed by IDWT using the appropriate wavelet and scaling bases. The resulting image is inverse color transformed from YCbCr to RGB if necessary. More detailed information on JPEG2000 can be found in [59, 15]. Furthermore, implement of the JPEG2000 algorithm is briefly explained in [60]

2.2.3 Adaptive Approach in Wavelet Based Image Coders

Lifting structure used in wavelets gives the opportunity of predicting high subband from low subband. So only transmitting the residuals of the high subband is enough. The block diagram of the lifting scheme without update stage is shown in Figure 2.7.

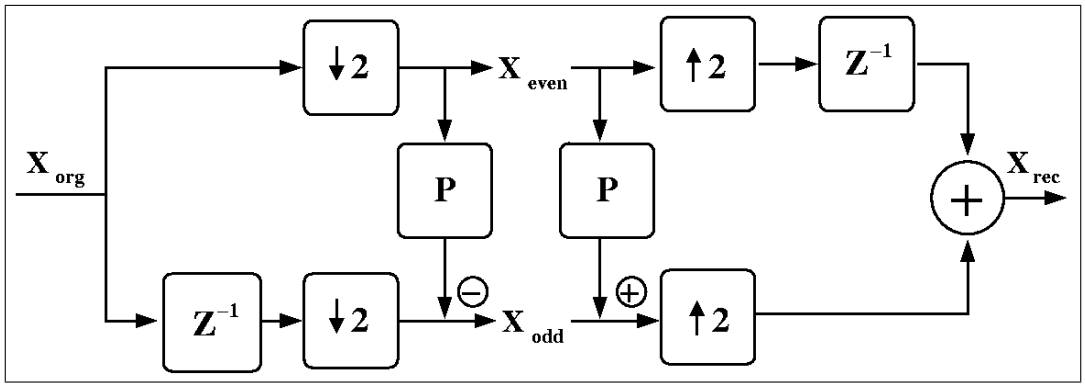


Figure 2.7: 1D lifting scheme without update stage.

Lifting structure also gives user the opportunity of implementing wavelet transform in a computationally efficient and separable way. Computational efficiency in wavelet transform is achieved by performing the filtering operation after the downsampling block. Separability of the lifting wavelet transform comes from the separability property if the used wavelet and scaling functions. Using lifting structure, an N-dimensional signal can transformed through each of its dimensions one-by-one separately. This can cause a drawback while exploiting the inter-sample correlation since the correlations in transform direction are taken care of. 2D signals like images can be a good example for the mentioned problem.

An image can be defined as $\mathbf{X}[n, m]$ where n and m are the indices of the pixels in horizontal and vertical directions respectively. Thus, $\mathbf{X}_L = \mathbf{X}[n, 2m]$ and $\mathbf{X}_H = \mathbf{X}[n, 2m+1]$ are low and high subbands in horizontal direction respectively. The pixels in $\mathbf{X}_H$ can be predicted from $\mathbf{X}_L$ by,

$$\hat{\mathbf{X}}_H[n, 2m+1] \;=\; (\mathbf{X}[n, 2m] \;+\; \mathbf{X}[n, 2m+2])/2. \tag{2.10}$$

Thus, the residual of high subband can be calculated as,

$$\mathbf{X}_{H\,\mathbf{e}}[n, 2m+1] \;=\; \mathbf{X}_H[n, 2m+1] \;-\; \hat{\mathbf{X}}_H[n, 2m+1]. \tag{2.11}$$

The problem here is that the predictions are always done in one direction (horizontal or vertical). Thus, when a diagonal edge is encountered, it can not be predicted efficiently by this scheme. Following the edge direction is a good idea while making predictions. What the adaptive wavelet transform does is adding flexibility to the prediction structure of the lifting wavelet transform. Thus, predictions in diagonal directions can also be done.

The predictor first calculates the derivatives along diagonal and horizontal directions as :

$$\begin{aligned}
\tilde{\mathbf{X}}_H[n, 2m+1] &= (\mathbf{X}[n, 2m] - \mathbf{X}[n, 2m+2])/2, \\
\tilde{\mathbf{X}}_{H135}[n, 2m+1] &= (\mathbf{X}[n-1, 2m] - \mathbf{X}[n+1, 2m+2])/2, \\
\tilde{\mathbf{X}}_{H225}[n, 2m+1] &= (\mathbf{X}[n+1, 2m] - \mathbf{X}[n-1, 2m+2])/2 \ . \qquad (2.12)
\end{aligned}$$

The minimum of $\tilde{\mathbf{X}}_H, \tilde{\mathbf{X}}_{H135}, \tilde{\mathbf{X}}_{H225}$ in Equation 2.12 gives the direction of the best prediction. Same approach can be used in the decoding stages so by adaptive inverse wavelet transform perfect reconstruction is also possible.

2.3 Connectivity-Guided Adaptive Wavelet Transform

The image-like representation of a 3D mesh structure can be composed using the operations in Section 2.1. At a first glance, the new data structure has no difference from an ordinary image so it can be coded using any image coders wanted. But unlike natural images there may be no correlation between neighboring pixels in our image-like mesh representation since neighboring pixels may not be coming from neighbouring vertices of the mesh.

Thus, instead of predicting non-zero pixels in our representation from their neighboring non-zero pixels, we here introduce an adaptive wavelet transform scheme which makes predictions using connectivity information. Ordinary wavelet transforms do not care about connectivity relationship. Side-by-side pixels are predicted from each other and then encoded. Our *Connectivity-Guided Adaptive Wavelet Transform* uses the connectivity information of the mesh and predict the pixels from its connected neighbors. In our approach we use lazy wavelet filter banks (Figure 2.5) and add a connectivity-guided adaptive prediction stage to it (Figure 2.8).

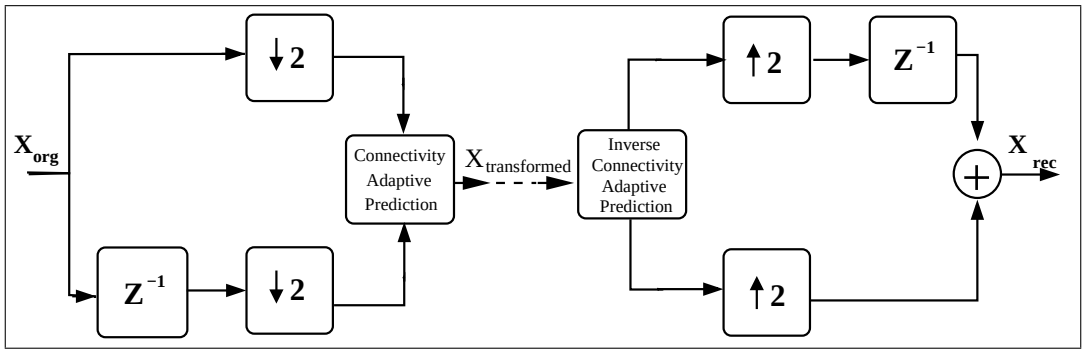


Figure 2.8: Lazy filter bank.

$\mathbf{I}_1[n_1, n_2]$ is the first channel image that stores the perpendicular distance between the corresponded vertex - pixel pair and $\mathbf{I}_2[n_1, n_2]$ stores the vertex indices. The lifting wavelet transform is implemented in a separable manner. Thus, both $\mathbf{I}_1$ and $\mathbf{I}_2$ are polyphased in the horizontal direction as,

$$\begin{aligned} \mathbf{I}_a[n_1, n_2] &= [\mathbf{I}_{a1}|\mathbf{I}_{a2}], \\ \mathbf{I}_{a1}[n_1, n_2] &= \mathbf{I}[n_1, 2n_2], \\ \mathbf{I}_{a2}[n_1, n_2] &= \mathbf{I}[n_1, 2n_2 + 1],\ a = 1,\ 2\ \ . \end{aligned} \tag{2.13}$$

Thus, $\mathbf{I}_{22}[n_1,\ n_2] = i,\ i \in \{1,\ \ldots,\ v\}$. Using the connectivity information we find a list of neighbors $nlist(j),\ j\ =\ 1,\ \ldots,\ v$ that holds the indices of the vertices connected to the vertex with index j. The predictions for $\mathbf{I}_{12}[n_1, n_2]$ values are done using $nlist_{valid}$, which is defined as,

$$nlist_{valid}(j) = nlist(j) \ \cap \ \mathbf{I}_{21}[n_1, n_2] \ . \tag{2.14}$$

List of vertex indices that are on image $\mathbf{I}_{22}$ is $list_{22}$. For each element k of $list_{22}$, a prediction should be found from $\mathbf{I}_{11}$ image. Valid neighbors of k can be found using Equation 2.14. So the prediction of vertex k is defined as :

$$\mathbf{I}_{k\,pred} = \frac{\sum_m (\mathbf{I}_{11}[n_1,\ n_2])}{m}, \tag{2.15}$$

where $\mathbf{I}_{21}[n_1,\ \ n_2] \ \ \in \ nlist_{valid}(k)$ and m is the number of the elements of $nlist_{valid}(k)$

$$\mathbf{I}_{new12}[n_1, n_2] = \mathbf{I}_{12}[n_1, n_2] - \mathbf{I}_{kpred}, \tag{2.16}$$

where $\mathbf{I}_{22}[n_1,\ n_2] \ = \ k$.

If no valid neighbors exist for a vertex, no estimation is carried out. Otherwise, a prediction is made for the value of the pixel. The same procedure is carried out between low and high subbands of the vertically transformed $\mathbf{I}_{11}$ image. The high pass part is only polyphased using lazy filters. So we have four small images at the end of each level. The inverse of the adaptive transform is also possible so that perfect reconstruction of the images is possible.

2.4 Connectivity-Guided Adaptive Wavelet Transform Based Compression Algorithm

An overview of our coding approach can be seen in Figure 2.9. A 3D mesh is represented by two image-like signals by applying the proposed mesh-to-image transform in Section 2.1. After transforming the projection image using connectivity-guided adaptive wavelet transform the data is quantized using a non-uniform

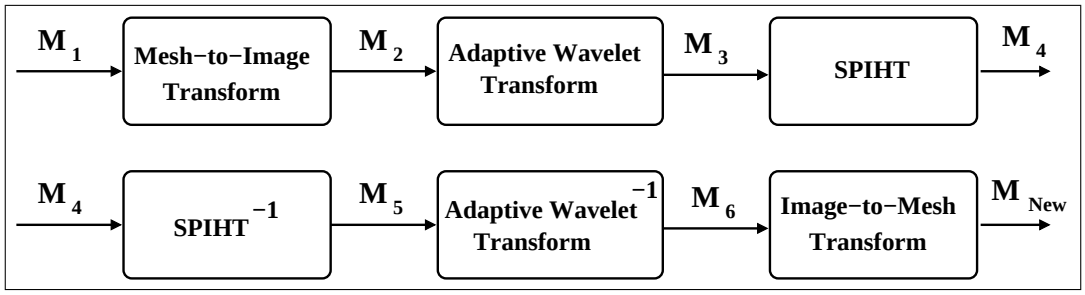


Figure 2.9: The block diagram of the proposed algorithm.

quantizer. After the non-uniform quantizer, the image is ready for SPIHT and JPEG2000 coding.

As mentioned in Section 2.1 geometry information of a mesh can be represented using two image-like signals by applying the proposed mesh to image transform. The correlations between projected mesh vertices are exploited using the connectivity-guided adaptive wavelet transform. A hierarchical representation for mesh vertices is achieved by this transform. Low subbands of the image contains more important vertices since the prediction of the higher subbands are made using those vertices. Before sending the image to SPIHT or JPEG2000 coders, it is quantized. Histogram of the images show that most of the pixel values are concentrated around zero. Thus, an 8 bit non-uniform quantization whose quantization steps are smaller around zero and larger on the sides is performed.

The quantized image is then fed into to the SPIHT or JPEG2000 coder. SPIHT coder hierarchically codes the image which means the lower subbands are at the initial parts of the code-stream and higher subbands follow them. So pruning the leading bits of the SPIHT stream causes a higher distortion in the reconstructed model than pruning the ending bits. By first reconstructing the model using leading bits and then refining it using the newly coming stream is possible. The decoding process is explained in Section 2.5.

When the quantized image is fed into JPEG2000 coder, it further quantizes the subbands of the tiles to the user specified levels. What makes JPEG2000

progressive is coding of the tiles. The tiles that corresponds to the lower subbands of the transformed image can be transmitted first and the reconstruction of the mesh can be done using them plus zeros padded instead of the other tiles. As the tiles corresponding to the higher subbands received, the reconstructed mesh can be refined. Here in this thesis this approach is not implemented. So our JPEG2000 coder is not a progressive coder.

After the bitstream is obtained by the SPIHT encoder, it should be arithmetically coded. We used *gzip* software as an arithmetic coder [61]. It is an implementation of the Lempel-Ziv coding algorithm [62]. For comparison purposes, both the original vertex list and the SPIHT bitstream are compressed using *gzip* software.

2.4.1 Coding of the Map Image.

The map images provide the projected vertex indices as its pixel values (Equation 2.9). The most important issue in the compression of these images is that it should be lossless. So no quantization can be done on the image. The pixel values has a wide range and they are equiprobable. Thus, a coding structure like Huffman of Lempel-Ziv is not appropriate for this kind of data.

For compression of these map images, a new algorithm which uses the principles of differential coding is proposed. The basis assumption of the proposed algorithm is near pixels represent near vertices. In the perfect case all K vertices of the mesh would be projected. Thus, a list of vertex locations on the image-like representation is created. The i^th entry of the list $LCoor(i)$ is defined as follows,

$$LCoor(i) = \begin{cases} [n_1, n_2], & \mathbf{I}[n_1, n_2] \; = \; i \; , \\ 0, & otherwise \end{cases} \tag{2.17}$$

where $i = 1, ..., v$. Then the $LCoor$ list is differentially coded. First the non-zero enteries of $LCoor$ are found as;

$$q(j) = \{i \mid LCoor(i) \neq [0,0] \; and \; j = 1, ..., G, \} \tag{2.18}$$

where G is the number of non-zero enteries of the $LCoor$ list. The $LCoor$ is updated as;

$$LCoor(q(j+1)) = LCoor(q(j+1)) - LCoor(q(j)). \tag{2.19}$$

By this way a predicted version of $LCoor$ is created whose mean is around 0 and variance is concentrated around the mean. Thus predicted version of $LCoor$ can be compressed more efficiently.

Then the $LCoor$ list is converted to a bitstream and sent to the receiver. What the receiver does is reversing the procedure. It finds again the first non-zero entry and it neighbors, inverses the predictions of the neighbors and then add those neighbors to the buffer. Processing the elements in the buffer one by one reverses the encoding process.

2.4.2 Encoding Parameters vs. Mesh Quality

Two issues defining the mesh quality are: **(i)** Length of the used bit-stream (SPIHT) & Quantization level for JPEG2000, **(ii)** Number of wavelet decomposition levels. Decreasing the length of the bitstream leads to more compression at the expense of higher distortion. Increasing the number of wavelet decomposition levels usually leads to higher compression ratios at the expense of more computational cost.

The distortion level of the reconstructed 3D mesh is measured visually or using some tools like METRO [63]. *Mean Square Error* (MSE) and *"Hausdorff Distance"* between the original and the reconstructed object are mostly used error measures in the literature.

The issue of how much of the SPIHT stream should be taken and what should be the quantization levels of JPEG2000 are closely related to the detail level parameter used in the orthogonal projection operation. If the detail level is low, the percentage of the used bit-stream must be increased to reconstruct the 3D mesh without much distortion.

Estimation of the number of levels of wavelet decomposition is easier than determining how much of the bitstream should be taken without introducing visual distortion. It is better to increase the number of scales in the wavelet decomposition as much as possible. This is because the data contains mostly insignificant pixels and as we get higher on the scales, the chance of having larger zero-trees is higher. Having larger zero-trees leads to better compression levels. However, increasing the decomposition level also increases the computational cost thus, adversely affects the performance.

2.5 Decoding

The 3D mesh is reconstructed using the SPIHT or JPEG2000 bitstream and some other side information, such as the vertex indexes (the second channel), the detail level used in the image-like representation. First the bitstream is transformed to image-like representation by decoding. Decoding process of SPIHT and JPEG2000 are explained in [49, 59] respectively. Then using the connectivity information of the mesh the inverse of the connectivity-guided adaptive wavelet transformation is applied to the image.

Finally, using the projected vertex coordinates $\check{\mathbf{v}}_i$ and the vertex indices, the image is back-projected to the 3D space. Since the only exact data available (except quantization) is the orthogonal component $\check{\mathbf{d}}_\mathbf{i}$ of the 3D mesh vertices, the mesh cannot be perfectly reconstructed. The real coordinates of the mesh are quantized to the grid point locations. The normalized 3D coordinates $\mathbf{v_i}$ of the mesh can be reconstructed without using any extra data.

From Equation 2.7 it is known that for each grid point the projected vertex is chosen among a set of vertices. The number of the elements of this set can be zero, one or more than one. Thus, some of the vertices can be coincided while they are projected onto the $\mathbf{n}$ plane. In projection operation, we choose one of the vertices from the set and discard coincided ones. Two methods are used to recover these vertices. One is to use a second plane to send the data of the coincided vertices; the second method is based on estimating the value of the vertex from its neighbors as in Equation 2.20. The connectivity list is used to find the neighbors of the lost vertex. Thus, the lost vertices can be predicted from their connected neighbors by,

$$\mathbf{v}_i = \frac{\sum_k \mathbf{v}_k}{k}, \tag{2.20}$$

where k is the number of elements of $n_{list}(i)$.

Chapter 3

Results from the Literature and Image-like Mesh Coding Results

This section is mainly consists of 3 parts: Section 3.1 gives the mesh coding results using the existing approaches, Section 3.2 gives the coding results of our algorithm,and Section 3.3 gives the comparisons between the our approach and the existing approaches.

3.1 Mesh Coding Results in the Literature

In Table 3.1, the compression rates of connectivity coders that were reviewed in Section 3.1 are given. The results in Table 3.2 show that valence based approaches [19],[16] give the best results [44]. Especially remeshing decreases the bit rate created by the valence based approaches drastically since most of the vertices of a regular mesh have a valence of six.

Since the method that is proposed in this thesis only compresses the geometry of the mesh, a connectivity compression algorithm is needed to compress the

Author	Name	Type	Bitrate (bpv)
Deering [3]	Generalized Triangle Mesh	Triangle Strips	$\sim$ 8-11
Taubin & Rossignac [18]	Topological Surgery	Dual Trees	~ 4
Gumbold & Strasser [64]	Cut Border Triangle Machine	Face Based	4.36
Rossignac [11]	Edgebreaker	Face Based	~ 3 (3.55 guaran.)
Isenburg & Snoeyink [30]	FaceFixer	Edge Based	~ 2.5
Touma & Gotsman [19]	The TG Coder	Valence Based	~ 2 ~ 0 when regular
Alliez & Desbrun [16]	Adaptive Valence Based	Valence Based-	~ 1.89 ~ 0 when regular

Table 3.1: Compression results for the single rate mesh connectivity coders in literature.

connectivity information of the mesh. Among the algorithms in Table 3.1, *Edgebreaker* [11] is used as the connectivity coder of the proposed method because it is one of the most widely used connectivity coding algorithm and has a public source code that can be used.

In Table 3.2 the compression rates of the existing single rate geometry coders are given. Corresponding quantization levels of the mesh vertices are given in the 4^{th} column of the table. Quantization level is an important concept while compressing geometry information of the mesh. From Table 3.2, it can be seen that by lowering the number of quantization levels (decreasing the number of bits) decreases the bit-rates. On the other hand, coarse quantization level results in more distortion in the reconstructed model.

In Table 3.3, the compression rates of existing progressive geometry coders are given. The best result is achieved by Khodakovsky et al. Their algorithm is based on the wavelet transformation of remeshed models. Like in connectivity coding, models that are remeshed to semi-regular or regular structures have lower

Author	Name	Bitrate	Quantization
		(bpv)	Levels
Touman & Gotsman [19]	Parallelogram	~ 20 ~ 26	12 Bits 14 Bits
Cohen-Or [33]	Multiway	~ 18 ~ 23	12 Bits 14 Bits
Isenburg & Alliez [65]	Polygonal Parallelogram	~ 16 ~ 20	12 Bits 14 Bits

Table 3.2: Compression results for the single rate mesh geometry coders in literature.

data rates. Due to the regular structures of the remeshed models, more accurate predictions can be done on those models so smaller residues result from the predictions of their vertices.

3.2 Mesh Coding Results of the Connectivity-Guided Adaptive Wavelet Transform Algorithm

Image coding results are significantly affected by the choice of wavelet basis. Therefore, wavelet transform basis to decompose the image-like meshes should be investigated. Among several wavelet bases like *lazy wavelet, daubechies-4, biorthogonal-4.4*, etc. lazy wavelet basis gave the best results. The reason is that most of the neighboring pixels in the image-like representation are not neighbors of each other in the original mesh representation. So a wavelet transform basis with a small support (lazy wavelet basis) gives better compression rates than those with larger support regions. This is because the image-like meshes contain isolated points.

In Tables 3.4 and 3.5 data sizes and distortions of the *sandal* and *cow* meshes that are compressed using SPIHT using wavelet transform of various basis, can

Author	Name	Bitrate (bpv)
Popovic & Hoppe [39]	PSC	over 35 bpv
Hoppe [21]	PM	about 35 bpv
Taubin [18]	PFS	slightly below 30 bpv
Pajarola & Rossignac [66]	CPM	about 22 bpv
Alliez & Desbrun [41]	VDC	14-20 bpv
Author	Name	Reconstruction Quality
Khodakovsky et. al. [43]	PGC	12 db better quality than CPM at the same bitrate
Khodakovsky & Guskov [44]	NMC	2-5 db better quality than PGC at the same bitrate
Morán & Garcia [17]	MG	between PGC and NMC
Gu et al. [5]	Geometry Images (GI)	3 db worse quality than PGC at the same bitrate
Praun & Hoppe [67]	Spherical GI	Better than PGC and GI worse than NMC

Table 3.3: Compression results for the progressive mesh coder in literature (Geometry + Connectivity).

be seen. Some of the corresponding meshes that are depicted in Table 3.4 and 3.5 can be seen in Figures 3.1 and 3.2.

Hausdorff distance metric is used for measuring the distortion between original and the reconstructed models. The *Hausdorff distance* between two given set of points $A = a_1, a_2, ..., a_n$ and $B = b_1, b_2, ..., b_n$ is :

$$d(A,b) = max\{\max_{a\in A}\min_{b\in B}|b-a|, \max_{b\in B}\min_{a\in A}|a-b|\}, \tag{3.1}$$

where $a - b$ is the Euclidean distance between two points. So the Hausdorff distance is the maximum distance of a set to the nearest element of the other set [68].

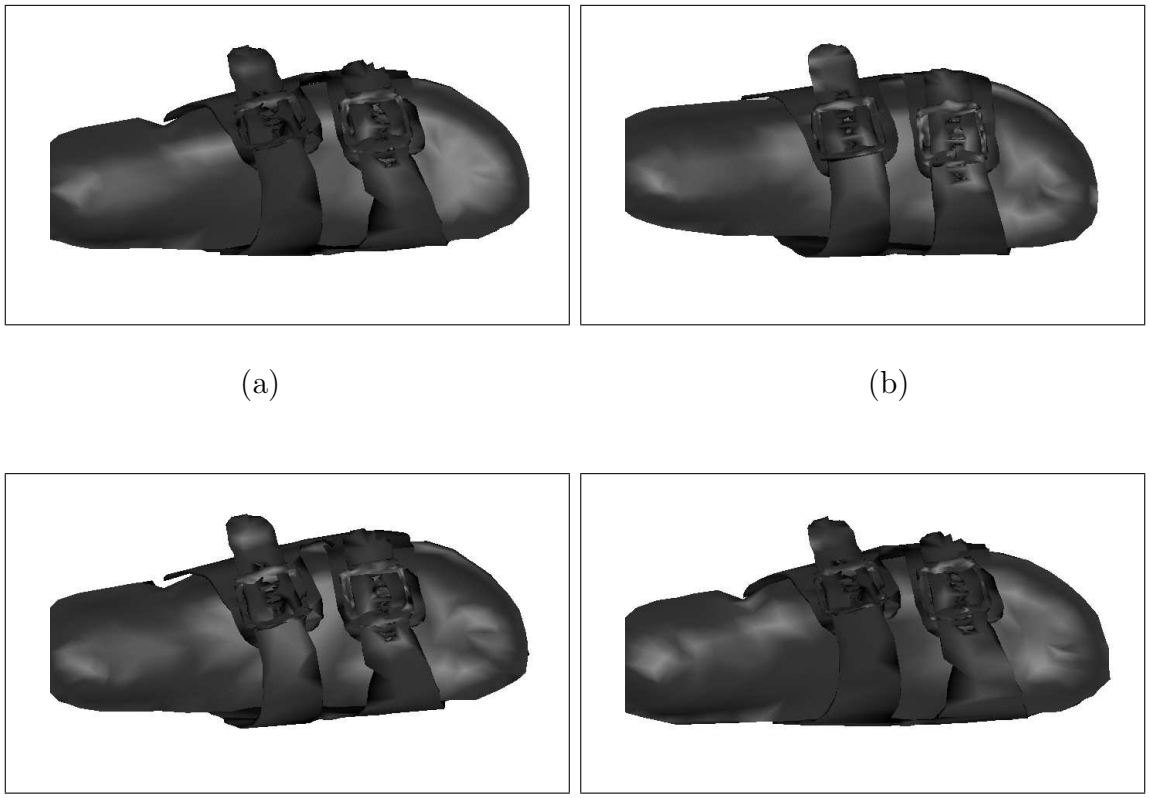
The factors that are affecting the Hausdorff distance between the original and the reconstructed model are: the *Detail level* and *the used bitstream. Detail level* parameter affects the size of the projection image and the closeness of the grid samples on the projection image. As the *Detail level* parameter increases

Filters	Bitstream Used (%)	Detail Level	Size (KB)	Org. - Quan. Hausdorff Dist.	Org. - Recons. Hausdorff Dist.
Lazy	40	3.0	6.39	0.022813	0.022830
Lazy	60	3.0	7.01	0.022813	0.0228224
Lazy	80	3.0	7.71	0.022813	0.022493
Lazy	60	4.5	7.84	0.021336	0.021347
Haar	40	3.0	8.51	0.022813	0.023225
Haar	60	3.0	10.2	0.022813	0.022827
Haar	80	3.0	12.1	0.022813	0.022825
Daubechies-10	60	3.0	13.0	0.022813	0.023813

Table 3.4: Compression results for the Sandal model.

Filters	Bitstream Used (%)	Detail Level	Size (KB)	Org. - Recons. Hausdorff Dist.
Without Prediction				
Lazy	40	3.0	9.51	0.014547
Lazy	60	3.0	10.4	0.014219
Haar	40	3.0	11.5	0.016085
Haar	60	3.0	13.8	0.014102
Haar	80	3.0	16.0	0.014011
Haar	100	3.0	18.0	0.014.28
Daubechies-4	40	3.0	12.0	0.018778
Daubechies-4	60	3.0	15.1	0.014661
Daubechies-4	80	3.0	18.2	0.014611
Daubechies-10	40	3.0	12.1	0.014194
Daubechies-10	60	3.0	15.2	0.045020
Biorthogonal-4.4	60	4.0	16.0	0.014549
Biorthogonal-4.4	80	3.0	18.3	0.023555
With Adaptive Prediction				
Lazy	20	3.0	9.64	0.025806
Lazy	30	3.0	10.6	0.013936
Lazy	60	3.0	12.6	0.014053
Lazy	80	3.0	13.6	0.014059
Lazy	40	5.0	19.0	0.007805
Lazy	60	5.0	21.2	0.007805
Lazy	80	5.0	23.4	0.007805
Lazy	60	7.0	22.1	0.007240
Lazy	80	7.0	23.9	0.007240

Table 3.5: Comparative compression results for the Cow model compressed without prediction and with adaptive prediction.



(a) (b)

(d) (e)

Figure 3.1: Reconstructed sandal meshes using parameters (a) lazy wavelet, 60% of bitstream, detail level=3; (b) lazy wavelet, 60% of bitstream, detail level=4.5; (c) Haar wavelet, 60% of bitstream, detail level=3; (d) Daubechies-10, 60% of bitstream, detail level=3.(Sandal model data is courtesy of Viewpoint Data Laboratories)

the projection image size increases and thus, the grid samples becomes closer in distance. *the used bitstream parameter* shows how much of the encoded bitstream is used in the reconstruction of the original model. Both the *Detail level* and *the used bitstream* parameters are linearly proportional to the size of the data used for reconstruction.

Using a standard wavelet filter basis including the lazy filter, interpixel correlations can not be exploited. Since the neighborhood concept between mesh vertices is related to the connectivity of the mesh, a wavelet transform that can adapt itself according to the connectivity of the mesh is needed. This idea leads

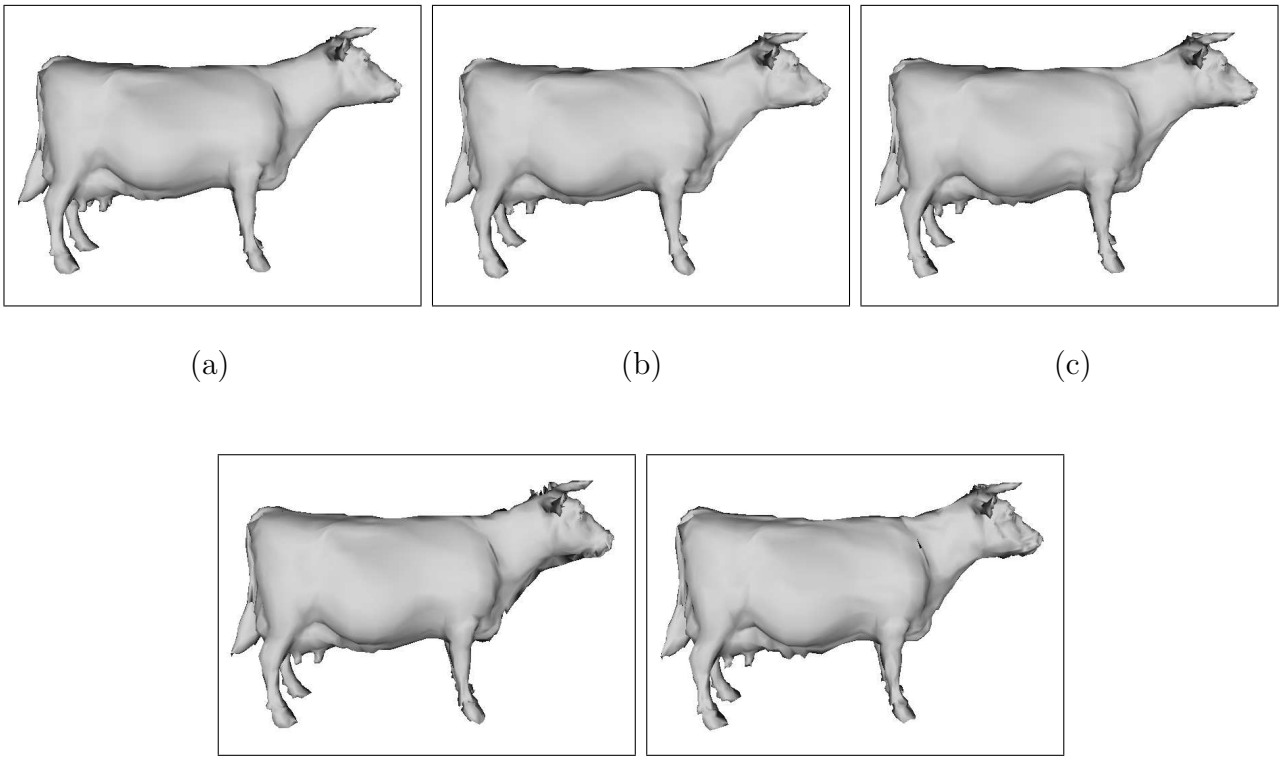


Figure 3.2: Meshes reconstructed with a detail level of 3 and 0.6 of the bitstream. (a) Lazy, (b) Haar, (c) Daubechies-4, (d) Biorthogonal4.4, and (e) Daubechies-10 wavelet bases are used.

to the implementation of the connectivity-guided adaptive wavelet transform. In this way pixels of the image-like representation are predicted from each other and small prediction errors are obtained. The comparative results of SPIHT compression of cow and lamp models, using non-adaptive and adaptive wavelet transforms are given in Tables 3.5 and 3.6.

Tables 3.5 and 3.6 show that adaptive wavelet structure produces better results than non-adaptive lazy filterbanks. Having the same data size adaptive structure causes less distortion in the reconstructed mesh. This means better compression for the same distortion rate. However, the computational power needed for the adaptive structure is higher since multiple searches in connectivity data must be performed.

Bitstream Used (%)	Detail Level	Size Non-adaptive (KB)	Size Adaptive (KB)	Org. - Recons. Hausdorff Dist. (Non-adaptive)	Org. - Recons. Hausdorff Dist. (Adaptive)
60	3.0	9.05	10	0.069620	0.023367
80	3.0	9.64	10.6	0.050859	0.019023
40	5.0	14.7	16.5	0.027902	0.018638
60	5.0	16.4	18.6	0.029870	0.018551
80	5.0	17.8	20.5	0.025991	0.018425
60	7.0	17.9	19.8	0.033990	0.011436
80	7.0	19.1	21.6	0.041133	0.012037

Table 3.6: Compression results for the Lamp model using lazy wavelet filterbank.

Figure 3.14 gives meshes reconstructed with and without using adaptive prediction for a qualitative comparison. Especially, the quality difference of the reconstructed meshes is noticeable at sharp features, such as the paws in the dragon mesh, and the base of the lamp mesh.

After embedding connectivity-guided adaptiveness in the algorithm, quantization is added to the algorithm. Quantization made the rate-distortion values better since; SPIHT and JPEG2000 adapt themselves better with the quantized data (1) and by using irregular quantization, the residuals part of the data is better represented (2). Figure 3.15 gives a qualitative comparison between the original and the reconstructed meshes which are compressed by SPIHT (a-b) and JPEG2000 (c). Meshes compressed with SPIHT are superior to JPEG2000 compressed meshes. This can also be noticed from Tables 3.7 - 3.8 which give rate distortion values for compressed Homer Simpson and 9 Handle Torus mesh models.

Captures from *MeshTool* [69] program shown in Figures 3.7, 3.8, 3.9, 3.10, 3.11, 3.12, 3.13,3.25, 3.26, 3.27, 3.28, 3.29, 3.3, 3.4, 3.5, 3.6, 3.21, 3.22, 3.23, 3.24, 3.29 visualize the distortion in the reconstructed models. *MeshTool* program also uses the Hausdorff distance as the distortion metric. The images referenced by (c) show the reconstructed model and the images referenced by (b) show

SPIHT Size-(KB)	Haus. Dist. (SPIHT)	JPEG2000 Size-(KB)	Haus. Dist. (JPEG2000)
4.07	0.060922	6.58	0.076215
4.37	0.033648	6.27	0.076107
4.67	0.019715	9.64	0.076488
4.96	0.013422	9.28	0.076374
5.55	0.015236	12.7	0.075922
6.76	0.005503	12.2	0.075699
7.92	0.005216	11.4	0.075680

Table 3.7: Compression results for the Homer Simpson model using SPIHT and JPEG2000. Hausdorff distances are measured between the original and reconstructed meshes.

SPIHT Size-(KB)	Haus. Dist. (SPIHT)	JPEG2000 Size-(KB)	Haus. Dist. (JPEG2000)
7.84	0.009638	13.6	0.036387
8.18	0.010951	14.0	0.036183
8.96	0.010699	14.1	0.036468
11.9	0.008904	16.7	0.036266
12.7	0.007685	16.9	0.036459

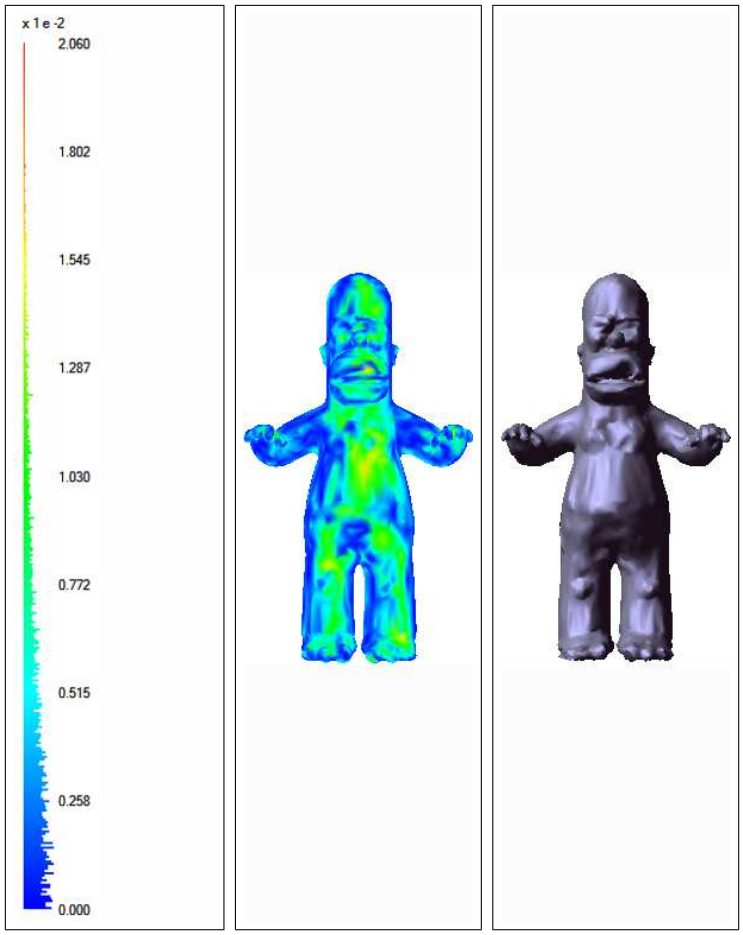
Table 3.8: Compression results for the 9 Handle Torus mesh model using SPIHT and JPEG2000. Hausdorff distances are measured between the original and reconstructed meshes.

the distortion levels as color levels on the original model. The graphs in those Figures show the histogram of the error on the original model. Blue is the lowest distortion level. As the distortion increases the color index goes from blue to green and then red. Red is the highest distortion level.

Figures 3.7, 3.8, 3.9, 3.10, 3.11, 3.12, 3.13 give the visualization of Homer model and Figures 3.25, 3.26, 3.27, 3.28, 3.29, give the visualization of 9-Torus mesh model reconstructed from different lengths of bitstream of the SPIHT code stream. As the length of the bitstream gets shorter, the distortion increases. The sharp features of the reconstructed model distort, as the length of the bitstream shortens.

Figures 3.3, 3.4, 3.5, 3.6 give the visualization of Homer model and Figures 3.21, 3.22, 3.23, 3.24, 3.29 give the visualization of the 9-Torus model reconstructed from different subband quantization levels. As the higher subbands are quantized more, the bitrate decreases but the distortion of the reconstructed model increases. The lower subbands of the image carries more information than the higher subbands because they are used for the prediction of the higher subbands. So fine quantization is done on low subbands and coarse quantization is done on the higher subbands.

The sandal and cow models are obtained from Viewpoint Datalabs Inc. and Matthias Müller (ETH Zürich), respectively. The original sandal model has a compressed data size of 31.7 KB and the cow model has a data size of 27.2 KB. The 9 Handle Torus model is obtained from *"http://www.ics.uci.edu/∼pablo/files/data/genus-non-0/9HandleTorus.ply"* and is composed of 9392 vertices with 165 KB compressed data size. The Homer Simpson model is obtained from *"INRIA Gamma Team Research database Website Collections"* and is composed of 4930 vertices with 98 KB data size.

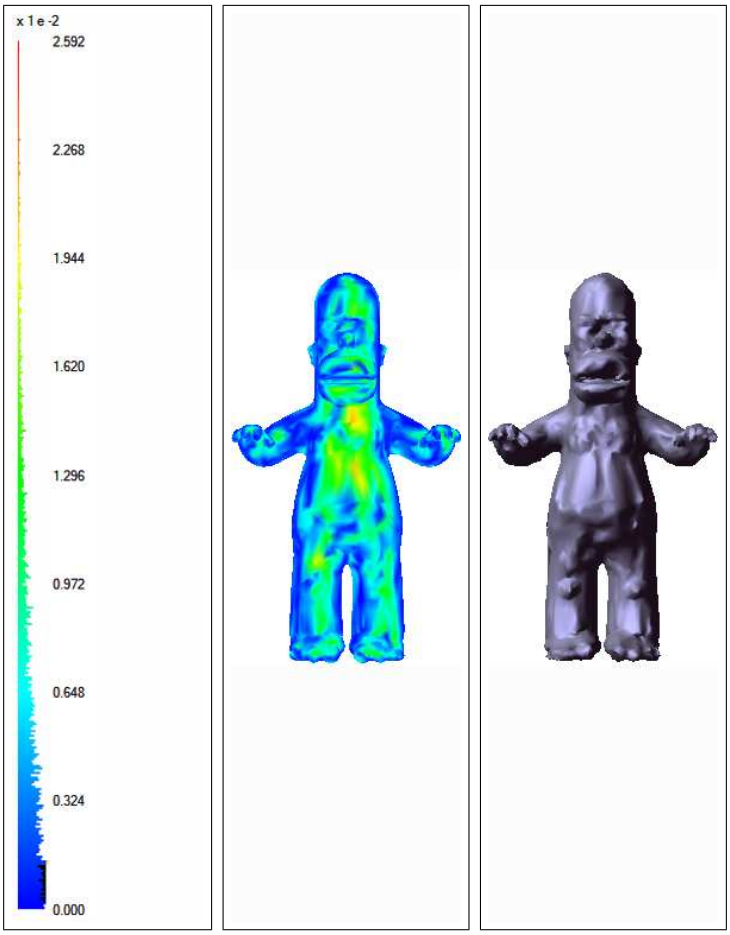


(a) (b) (c)

Figure 3.3: Homer Simpson model compressed with JPEG2000 to 6.58 KB (10.7 bpv).

3.3 Comparisons

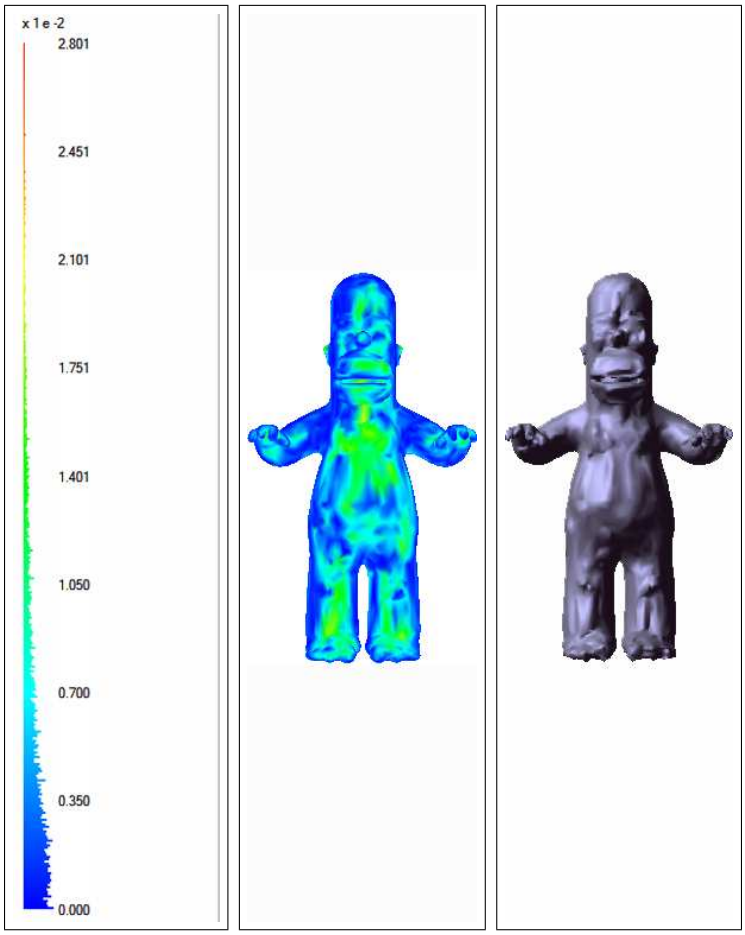
In single rate compression schemes, the mesh data is compressed before transmission and then all the data is sent. In progressive compression schemes, the mesh data is compressed during transmission and sent progressively. First low-resolution data is decoded and then the decoded model is updated to a higher resolution using newly arriving data. In our approach, we use the prior compression property of single rate compression, and updateable decoded model property of progressive compression.



(a) (b) (c)

Figure 3.4: Homer Simpson model compressed with JPEG2000 to 6.27 KB (10.1 bpv).

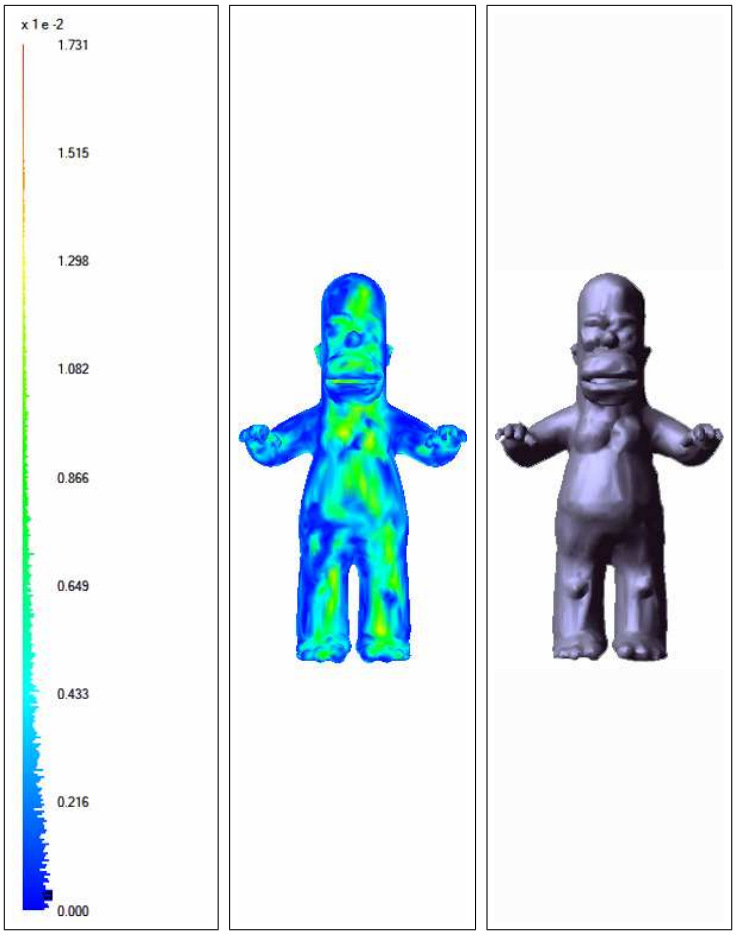
The bitstream created by SPIHT coding has an hierarchical structure. Thus, after receiving the first l coefficients of the bitstream where l is smaller than the total length of the bitstream L, the decoder reconstructs a lower resolution representation of the mesh. Once a mesh is formed using the first l coefficients, the decoder can update the mesh using the remaining bitstream elements in a progressive manner. The mesh can be perfectly reconstructed once the whole bitstream is received. This property makes the proposed SPIHT approach a progressive mesh compression technique.



(a) (b) (c)

Figure 3.5: Homer Simpson model compressed with JPEG2000 to 9.28 KB (15 bpv).

Most of the progressive mesh compression algorithms use edge collapses at the encoder side and vertex splits at the decoder side. A typical low resolution version of a given mesh has a smaller number of vertices and larger triangles than the original mesh [48]. One cannot retrieve a lower resolution mesh structure from the bitstream in the proposed algorithm because the SPIHT encoder compresses the wavelet domain data by taking advantage of the multiscale correlation between scales. Because of this the proposed compression technique is similar to single rate coding schemes. All the vertices can be recovered from a part of the bitstream. The degradation in the quality comes from the distortion due to the



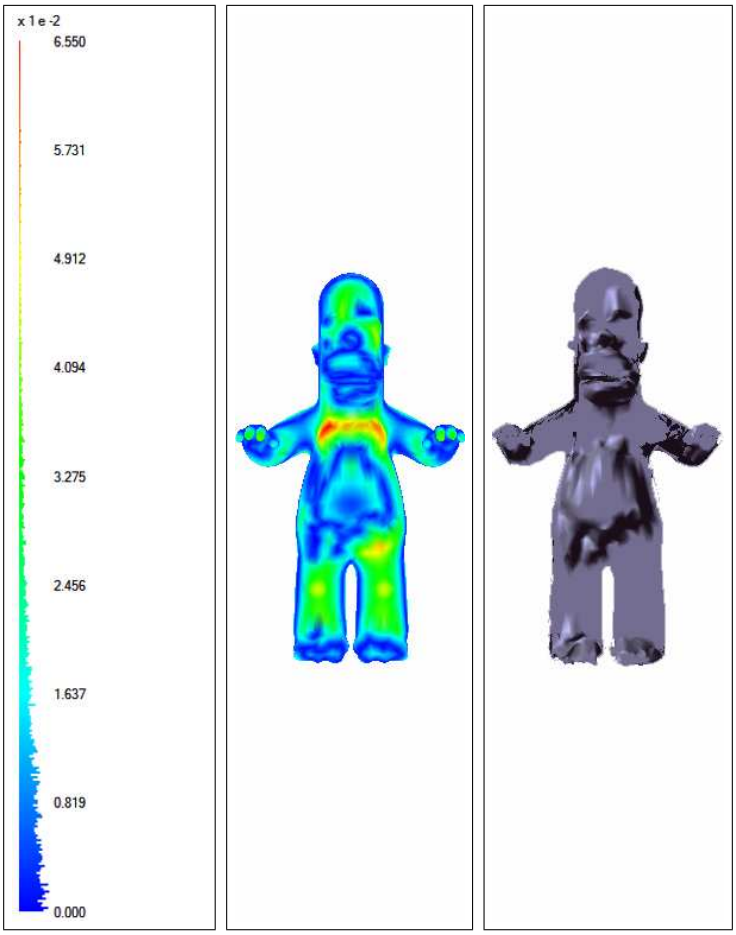
(a) (b) (c)

Figure 3.6: Homer Simpson model compressed with JPEG2000 to 12.7 KB (20.6 bpv).

inexact positioning of the vertices. These features of the proposed algorithm are demonstrated in Figure 3.16 for various resolution degradation levels.

The difference between two resolutions in progressive mesh compression is represented by using a sequence of edge collapses or vertex splits [66]. In the proposed SPIHT algorithm, the difference between two resolutions is represented by using the parts of the bitstream that fine tune the positions of the mesh vertices.

On the other hand, our JPEG2000 algorithm is not a progressive encoder as it is implemented here. But different resolutions of the mesh model can be

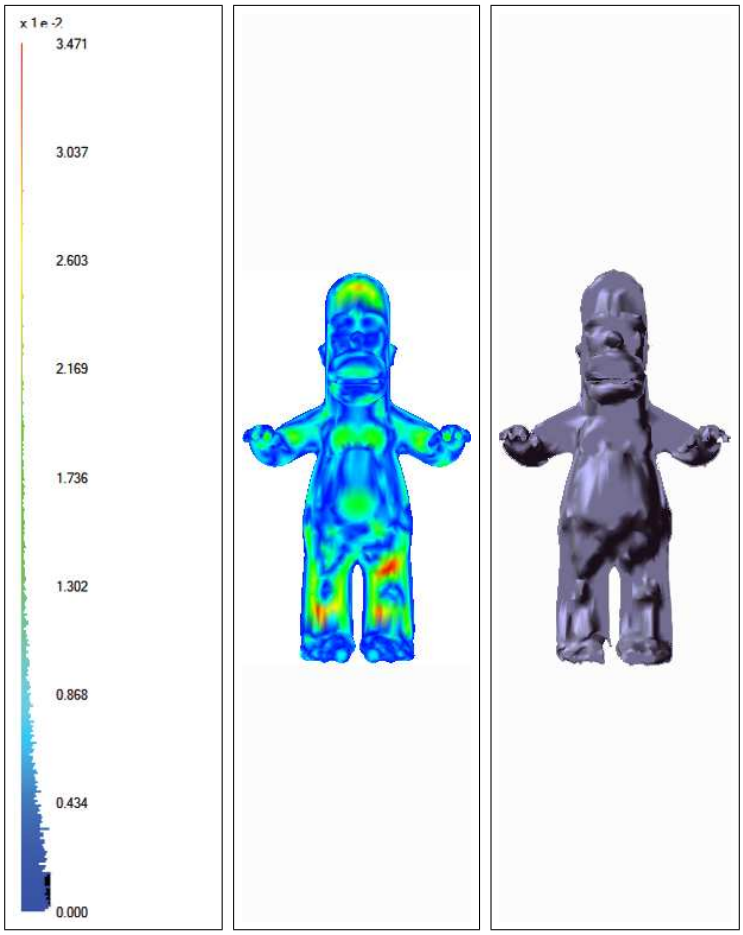


(a) (b) (c)

Figure 3.7: Homer Simpson model compressed with SPIHT to 4.07 KB (6.6 bpv).

reconstructed by using different quantization levels in JPEG2000. In the wavelet subband decomposition high bands are suppressed due to coarse quantization so that lower resolution mesh models are established. Using the scheme, which is explained in Subsection 2.4, this approach can be converted to a progressive mesh coding algorithm.

Another image-like coding technique for mesh models is also mentioned in [35]. The images are created using parameterization of the mesh. Parameterizing a mesh is a very computationally hard job to do. Also before parameterization the mesh model should be cut and opened which are also hard tasks

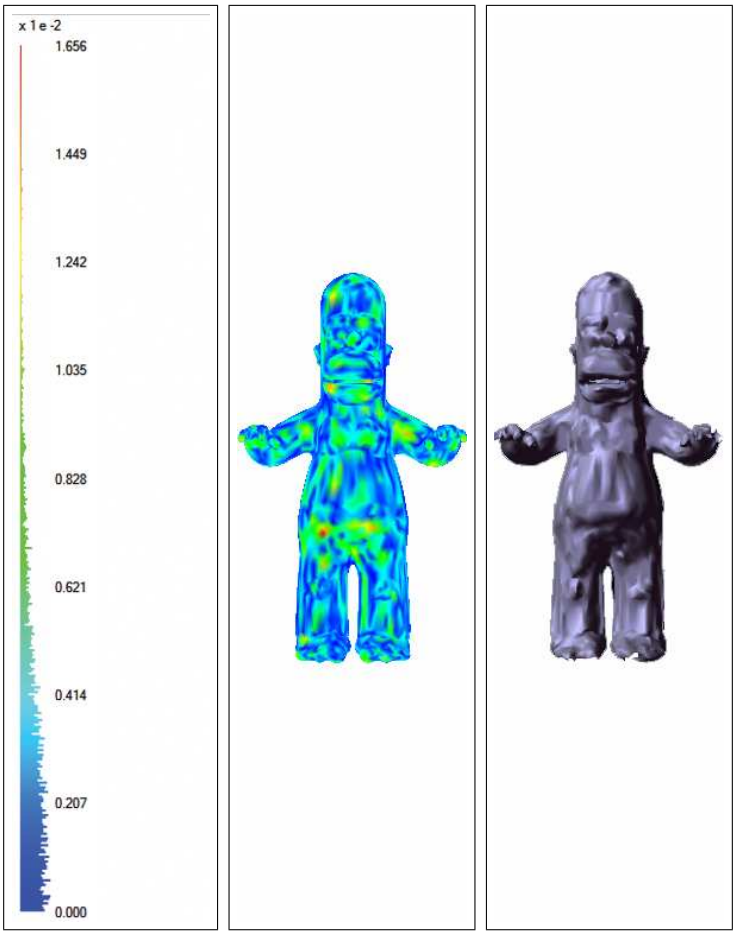


(a) (b) (c)

Figure 3.8: Homer Simpson model compressed with SPIHT to 4.37 KB (7.1 bpv).

to do for complex models. In contrary to parameterization, our projection operation is a easy to implement and computationally easy job to do. It do not need the to be cut and opened. But the drawback of projection operation is that it does not guarantee all the vertices to be projected on the image-like representation. But using connectivity based interpolation, the unprojected vertices can be interpolated from the know vertex locations.

Also other wavelet based mesh coders, like PGC in [44], do exist in the literature. PGC is one of the most successful mesh coder in literature. From Figure 3.17 it can be seen that using only the % 15 (b) of a stream of 37.7KB results in a visually very good result. As the percentage of the used stream

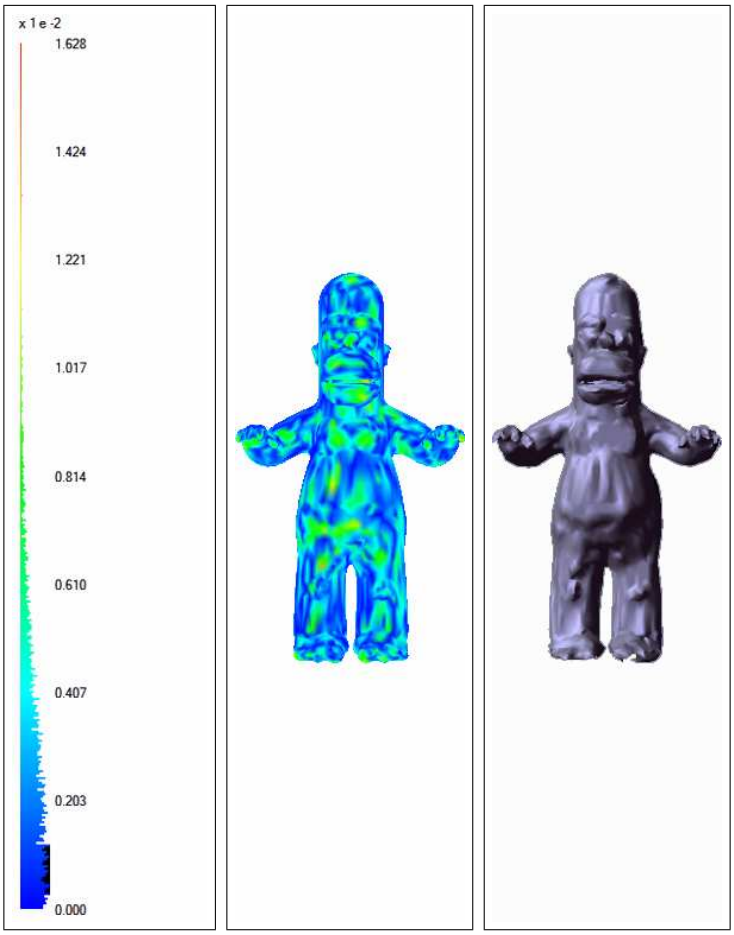


(a) (b) (c)

Figure 3.9: Homer Simpson model compressed with SPIHT to 4.67 KB (7.6 bpv).

increases the quality of the model increases too (Figure 3.17 (a)-(f)). But the problem with PGC is that it can work only on semi-regular or regular meshes. Thus, before using wavelet transformation directly on meshes, the mesh model must be remeshed so that a semi-regular or regular mesh structure is established. Remeshing is also a computationally hard task; especially for large models. On the other hand, our scheme can be applied to any mesh regardless of its regularity.

In Figures 3.18,3.19,3.20 a visual comparison between MPEG mesh coder and the proposed SPIHT based mesh coder is given. As it is shown in Table 3.9 the proposed algorithm gives comparable results with the MPEG mesh coder. When the same *mean distance* between the original and reconstructed models



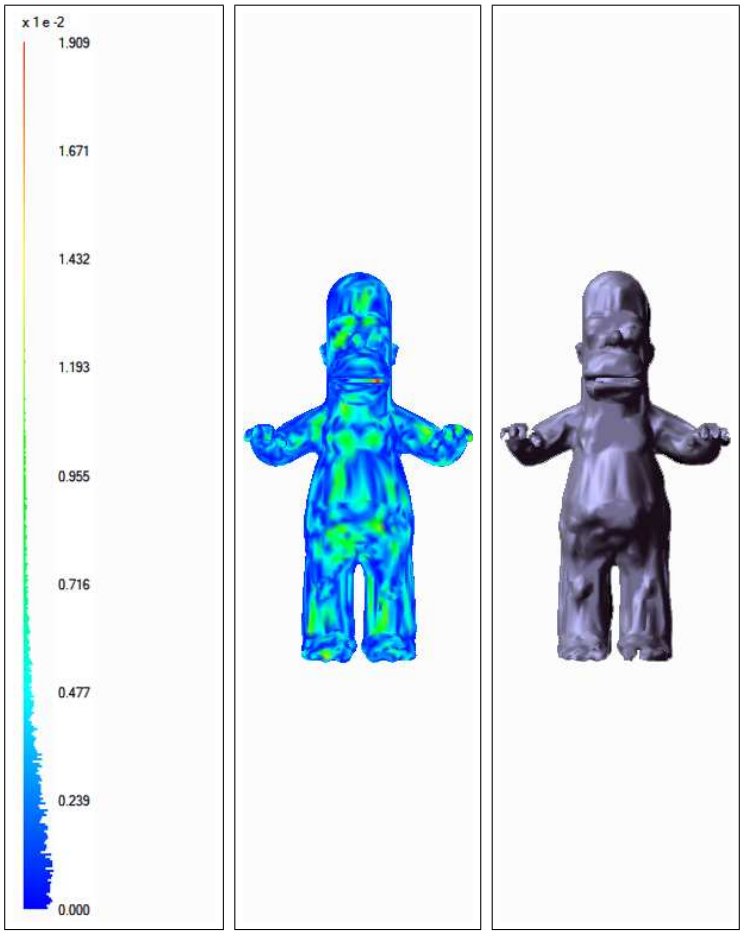
(a) (b) (c)

Figure 3.10: Homer Simpson model compressed with SPIHT to 4.96 KB (8 bpv).

taken into account,the data size of the proposed coder's is superior to the MPEG coder. Error in the high pass parts of the models is due to the lost vertices in the projection operator. If a better projection operator can be defined, those error can be corrected and much better results can be obtained.

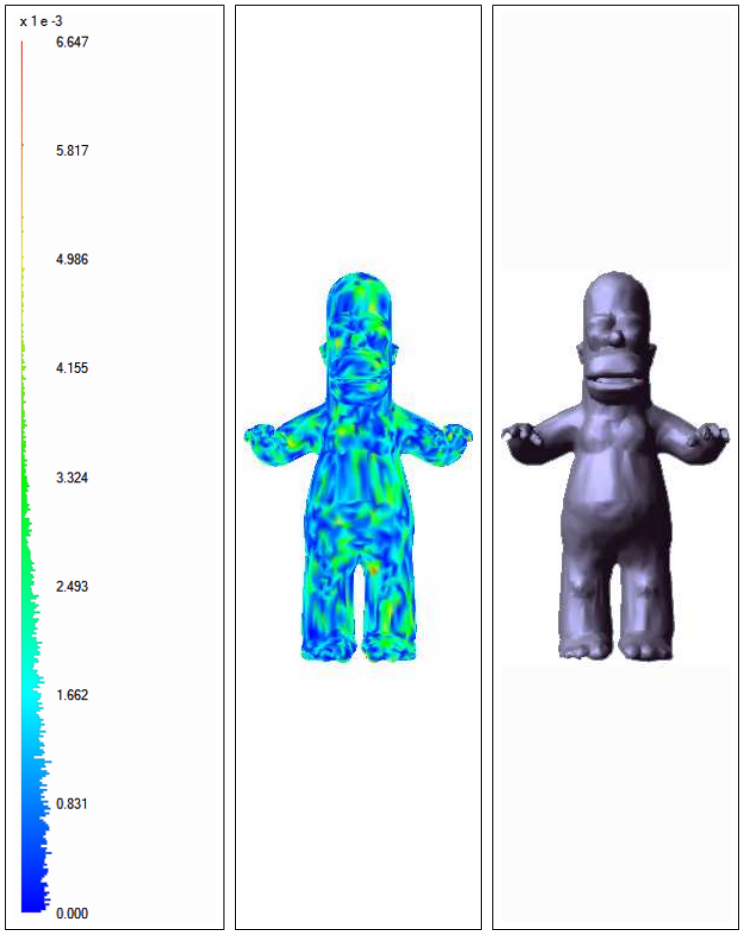
Model	Compression Algorithm	Data Size-(KB)	Max Distance Hausdorf	Mean Distance
Homer	MPEG	41.8	0.002645	0.00066
Homer	SPIHT	9.41	0.003704	0.00060
Homer	SPIHT	7.92	0.005216	0.00093
Inek	MPEG	26.1	0.001780	0.00068
Inek	SPIHT	7.25	0.005631	0.00041
Lamp	MPEG	36	0.43	0.1
Lamp	SPIHT	4.77	0.01468	0.00170
9Han.Torus	MPEG	82.8	0.001563	0.0005976
9Han.Torus	SPIHT	12.7	0.009797	0.000927
Sandal	MPEG	22.7	0.001904	0.000743
Sandal	SPIHT	5.91	0.007705	0.000273
Sandal	SPIHT	4.2	0.020076	0.000788
Dance	MPEG	55.4	0.002007	0.000673
Dance	SPIHT	17.3	0.003393	0.000326
Dance	SPIHT	12.1	0.009140	0.00106
Dragon	MPEG	43.1	0.001473	0.000557
Dragon	SPIHT	7.18	0.05672	0.00192

Table 3.9: Comparative results for the Homer,9 Handle Torus, Sandal, Dragon, Dance mesh models compressed using MPEG and SPIHT mesh coders. Hausdorff distances are measured between the original and reconstructed meshes.



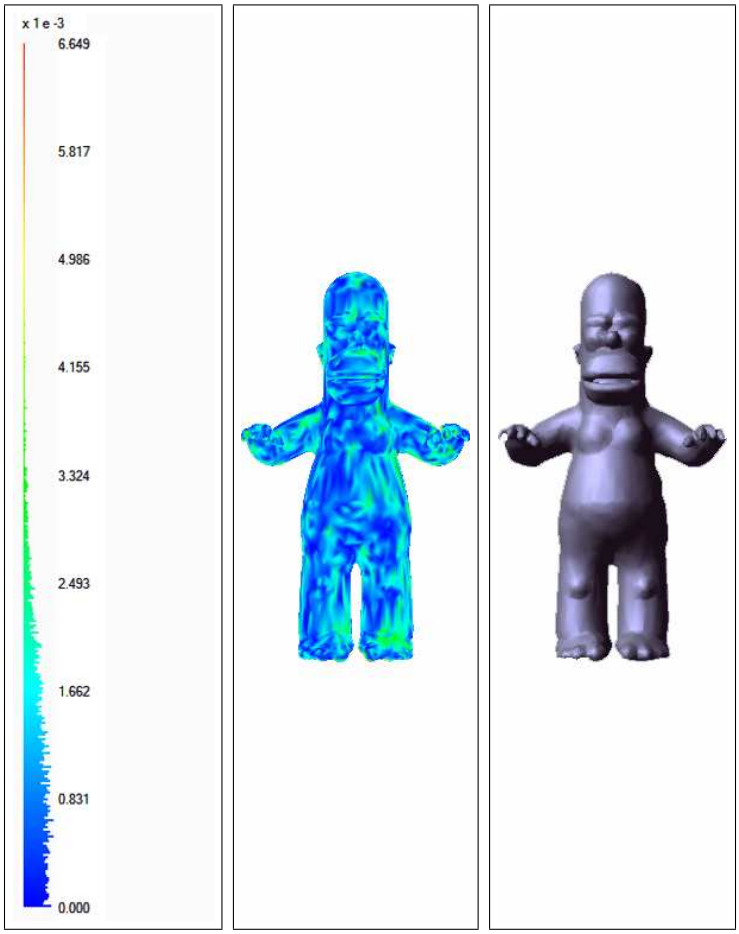
(a) (b) (c)

Figure 3.11: Homer Simpson model compressed with SPIHT to 5550 KB (9 bpv).



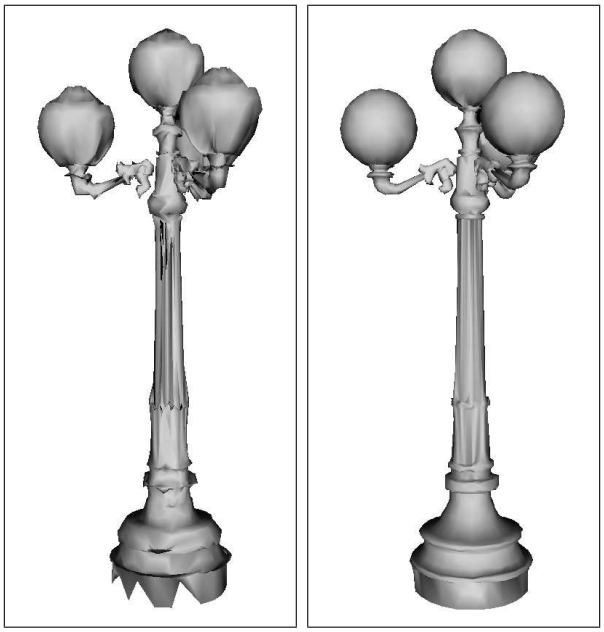
(a) (b) (c)

Figure 3.12: Homer Simpson model compressed with SPIHT to 6.76 KB (11 bpv).

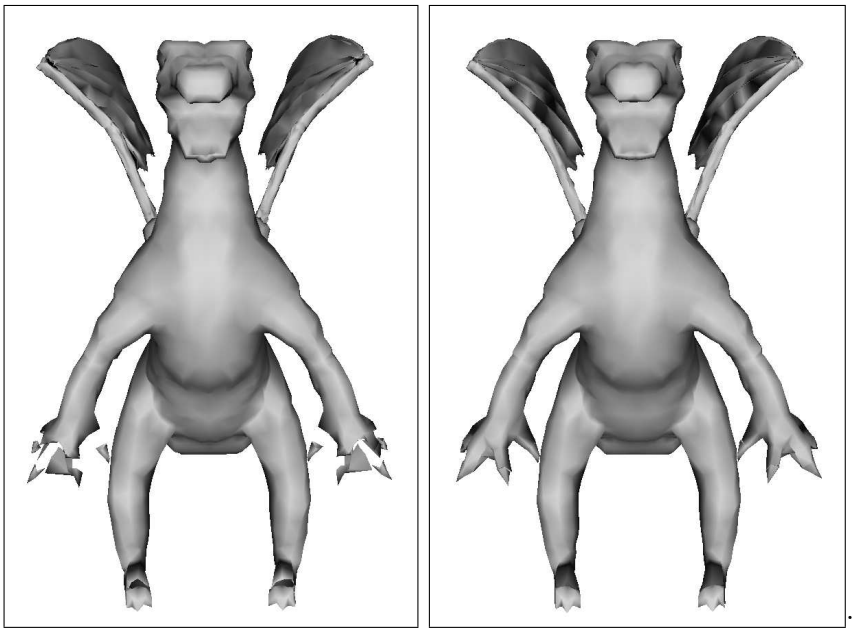


(a) (b) (c)

Figure 3.13: Homer Simpson model compressed with SPIHT to 7.92 KB (12.8 bpv).

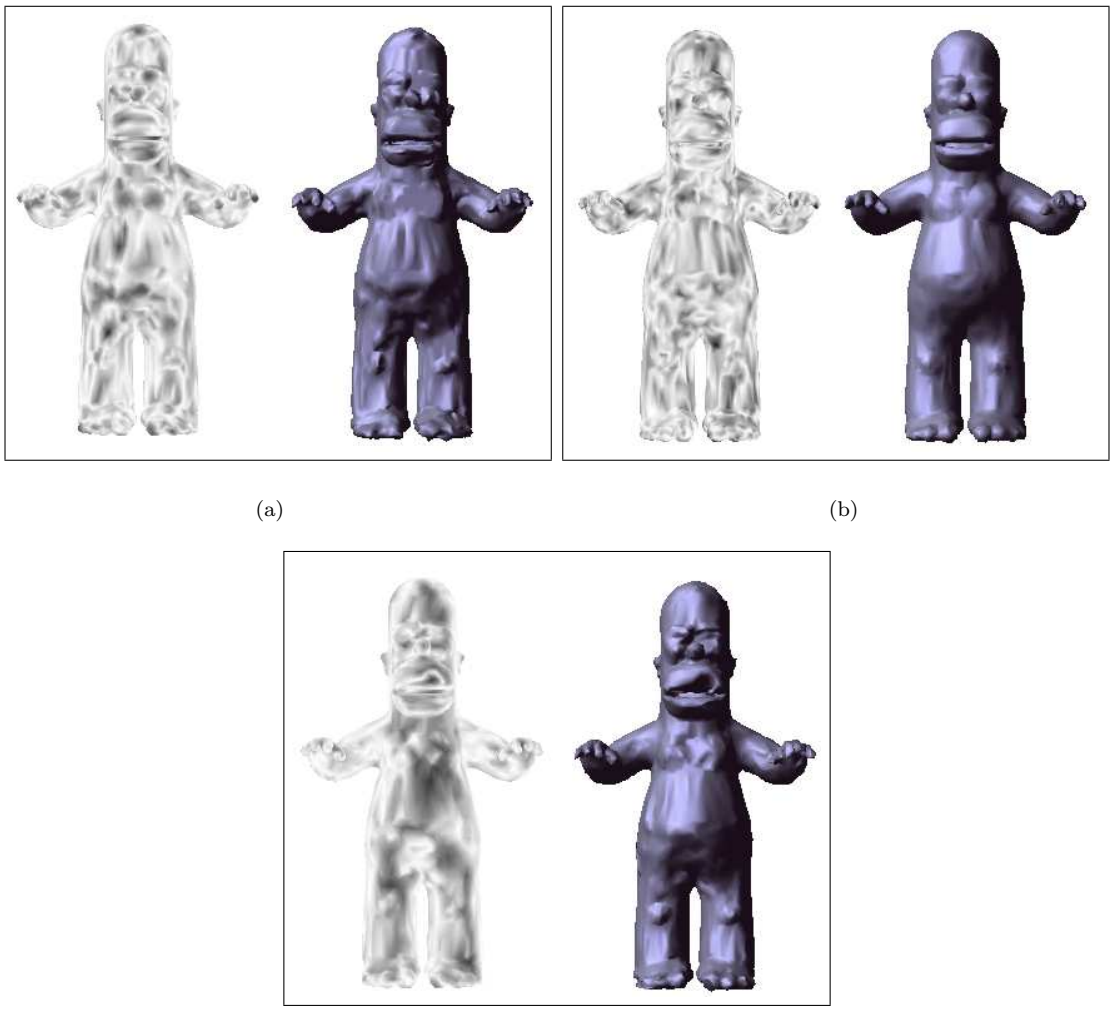


(a) (b)



(c) (d)

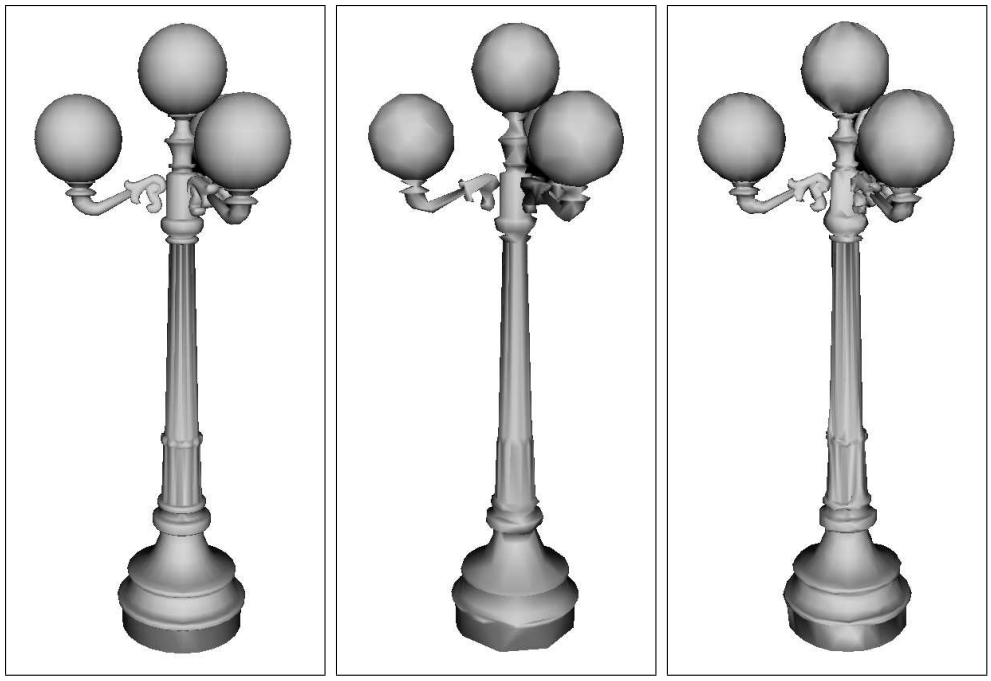
Figure 3.14: The qualitative comparison of the meshes reconstructed with without prediction (a and c) and adaptive prediction (b and d). Lazy wavelet basis is used. The meshes are reconstructed using 60% of the bitstream with detail level=5 in the Lamp model and 60% of the bitstream with detail level=5 in the Dragon model. (Lamp and Dragon models are courtesy of Viewpoint Data Laboratories)



(a) (b)

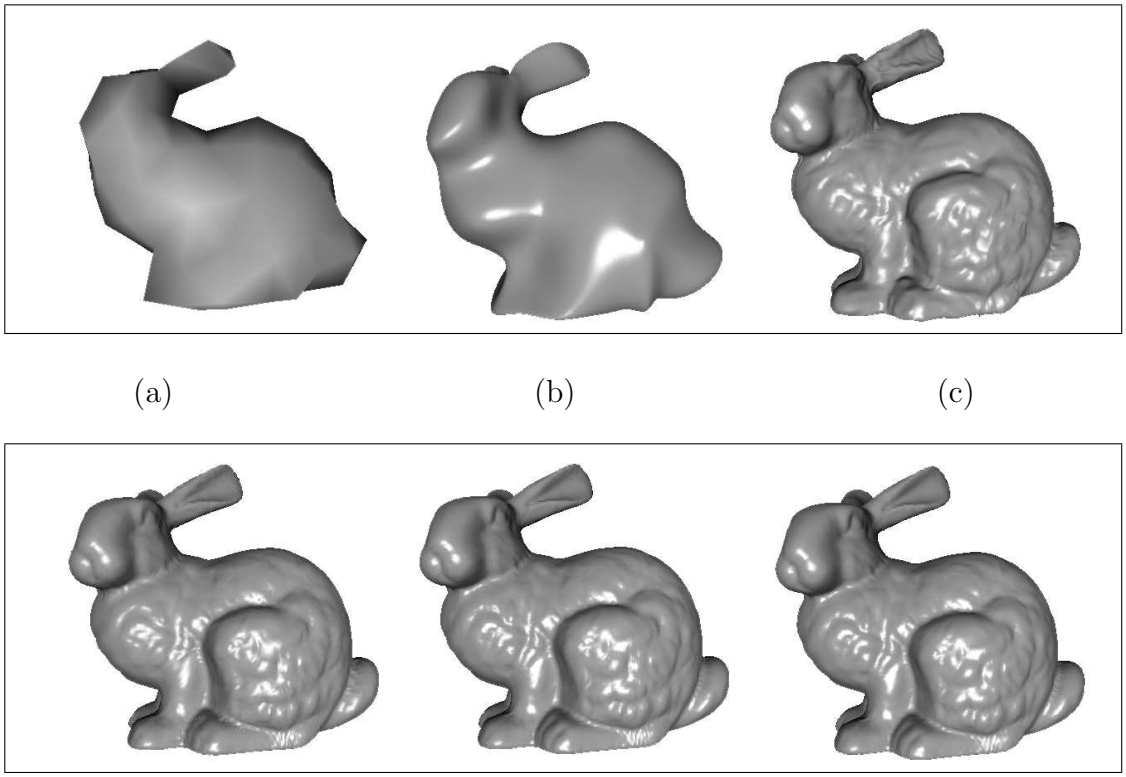
(c)

Figure 3.15: Distortion measure between original (images at left side of the figures) and reconstructed Homer Simpson mesh models using MeshTool [69] software. (a) SPIHT at 6.5 bpv; (b) SPIHT at 11 bpv; (c) JPEG2000 at 10.5 bpv. The grayscale colors on the original image show the distortion level of the reconstructed model. Darker colors mean more distortion.



(a) (b) (c)

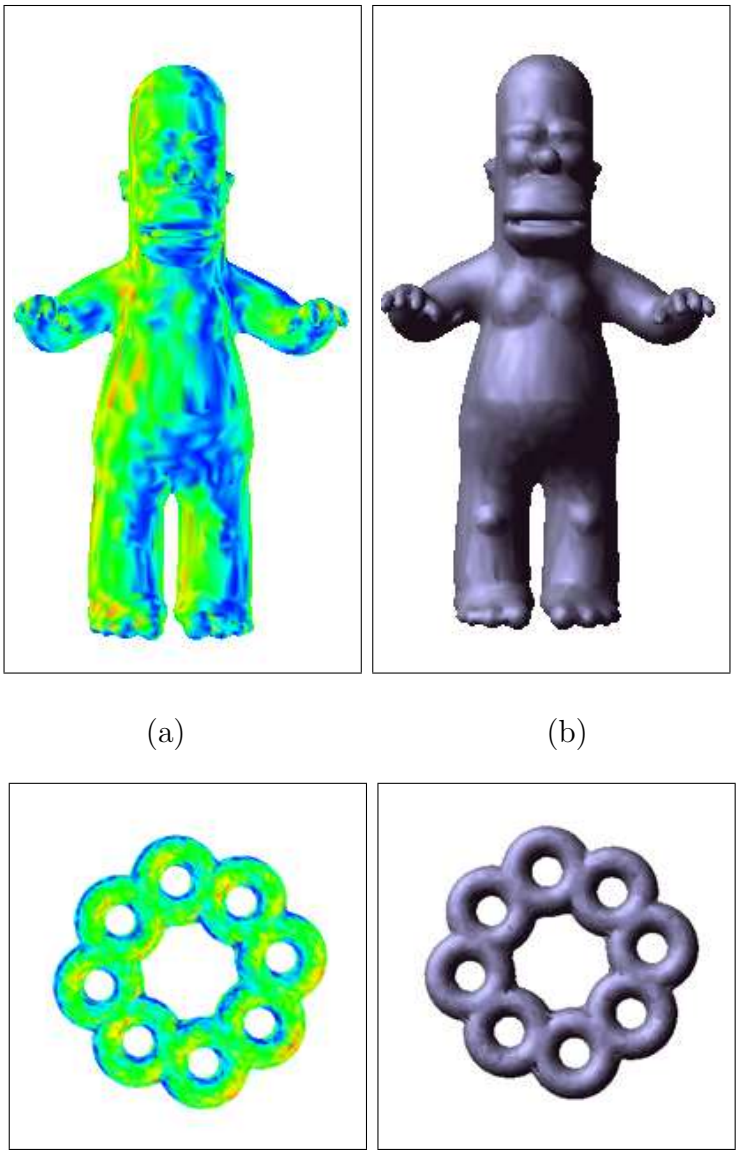
Figure 3.16: Comparison of our reconstruction method with Garland's simplification algorithm [48] (a) Original mesh; (b) simplified mesh using [48] (the simplified mesh contains 25% of the faces in the original mesh); (c) mesh reconstructed by using our algorithm using 60% of the bitstream.



(a) (b) (c)

(d) (e) (f)

Figure 3.17: (a) Base mesh of Bunny model composed by PGC algorithm (230 faces); (b) Model reconstructed from 5% of the compressed stream (69, 967 faces); (c) Model reconstructed from 15% of the compressed stream (84, 889 faces); (d) Model reconstructed from 50% of the compressed stream (117, 880 faces); (e) Model reconstructed from 5% of the compressed stream (209, 220 faces); (f) Original Bunny mesh model (235, 520 faces). The original model has a size of 6 MB and the compressed full stream has a size of 37.7 KB.



(a) (b)

(c) (d)

Figure 3.18: Homer and 9 Handle Torus models compressed using MPEG mesh coder. The compressed data sizes are 41.8 KB and 82.8 KB respectively. Figures on the left side show the error of the reconstructed model with respect to the original one. Reconstructed models are shown on the right side.

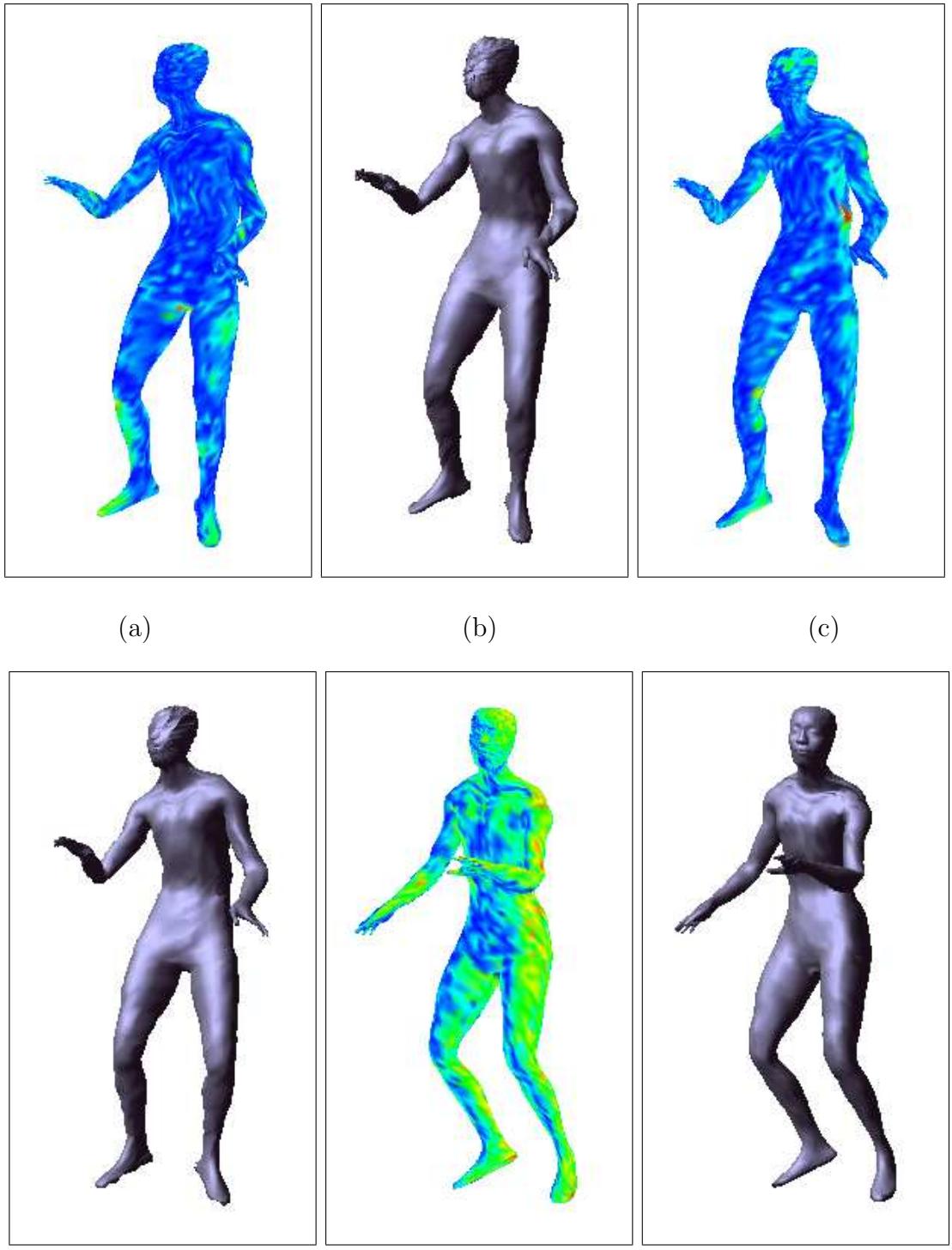
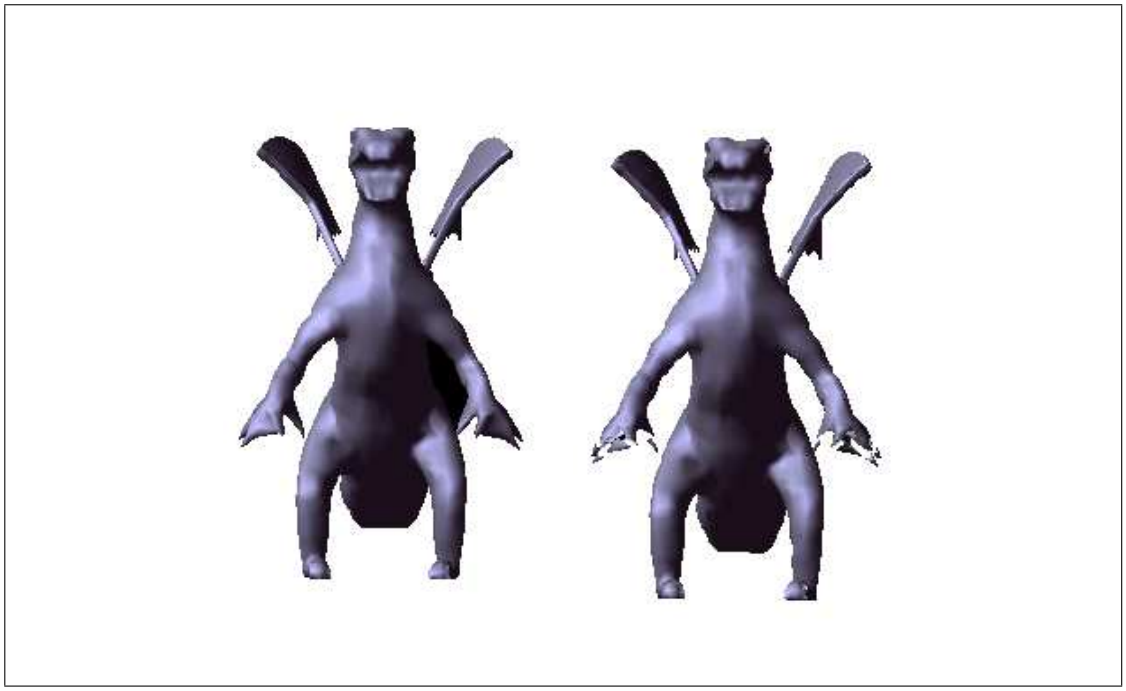
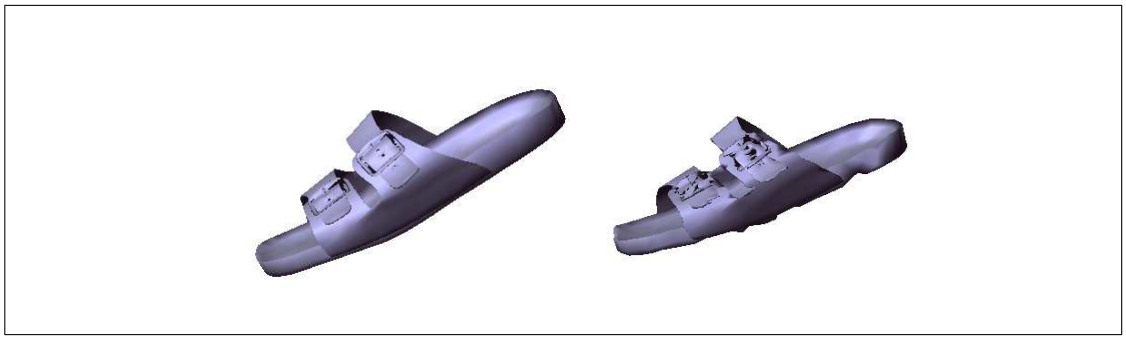


Figure 3.19: The error between the original dancing human model and reconstructed dancing human models compressed using SPIHT 13.7 bpv (a) and 9.7bpv (c) respectively. (b) and (d) show the reconstructed models. The error between the original dancing human model and reconstructed dancing human models compressed using MPEG at 63 bpv (e). (f) the reconstructed models.

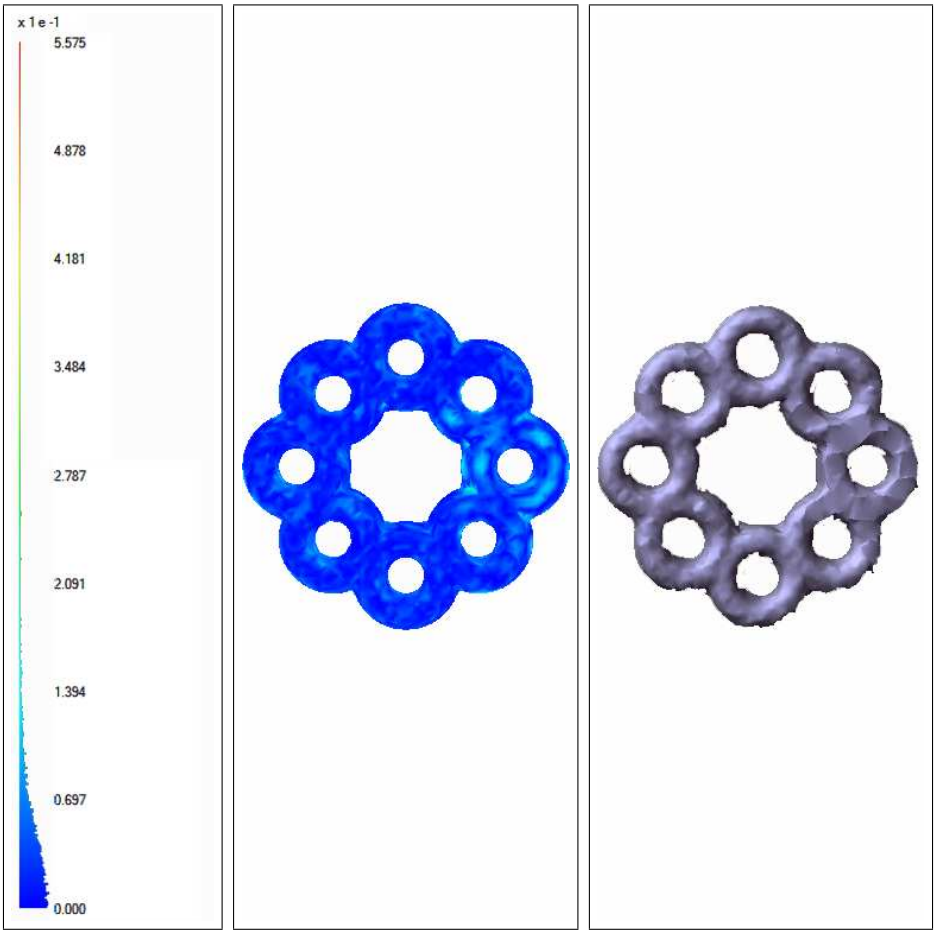


(a) (b)



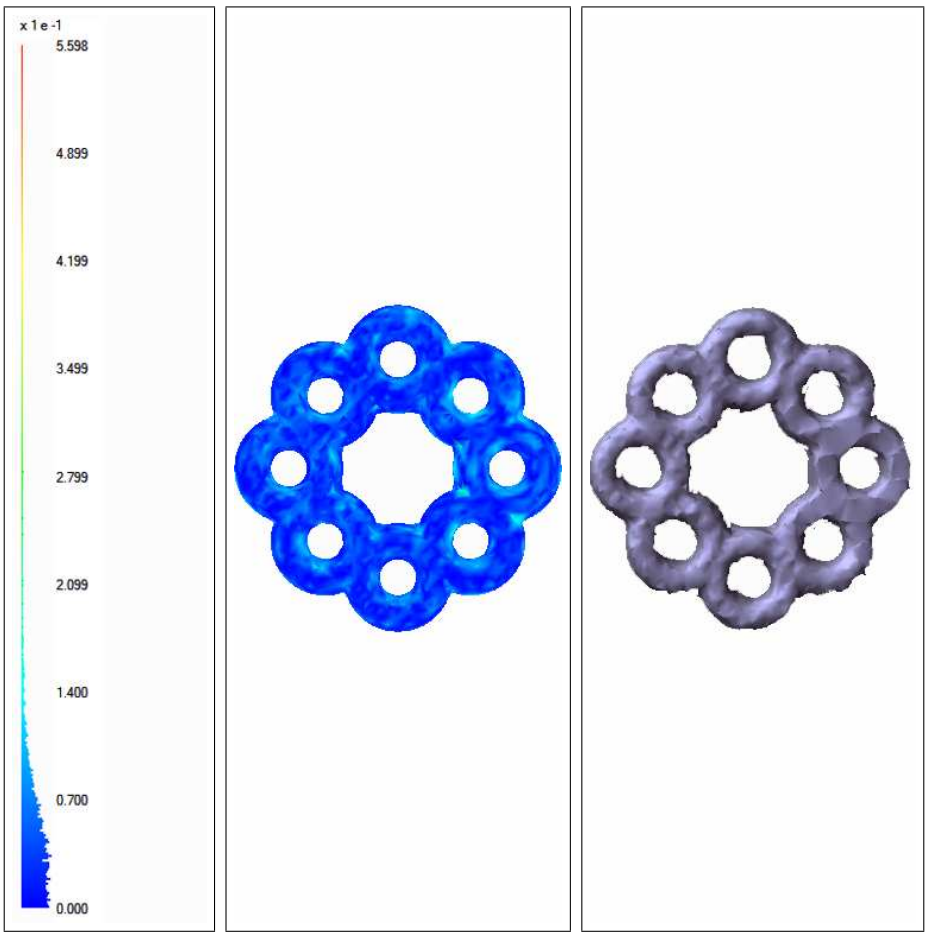
(c) (d)

Figure 3.20: (a) Dragon (5213 vertices) and (c) Sandal (2636 vertices) models compressed using MPEG mesh coder.(b) Dragon (5213 vertices) and (d) Sandal (2636 vertices) models compressed using The proposed SPIHT coder. Compressed data size are 43.1 KB and 10.4 KB, respectively for Dragon model and 22.7 KB and 2.77 KB respectively for Sandal model.



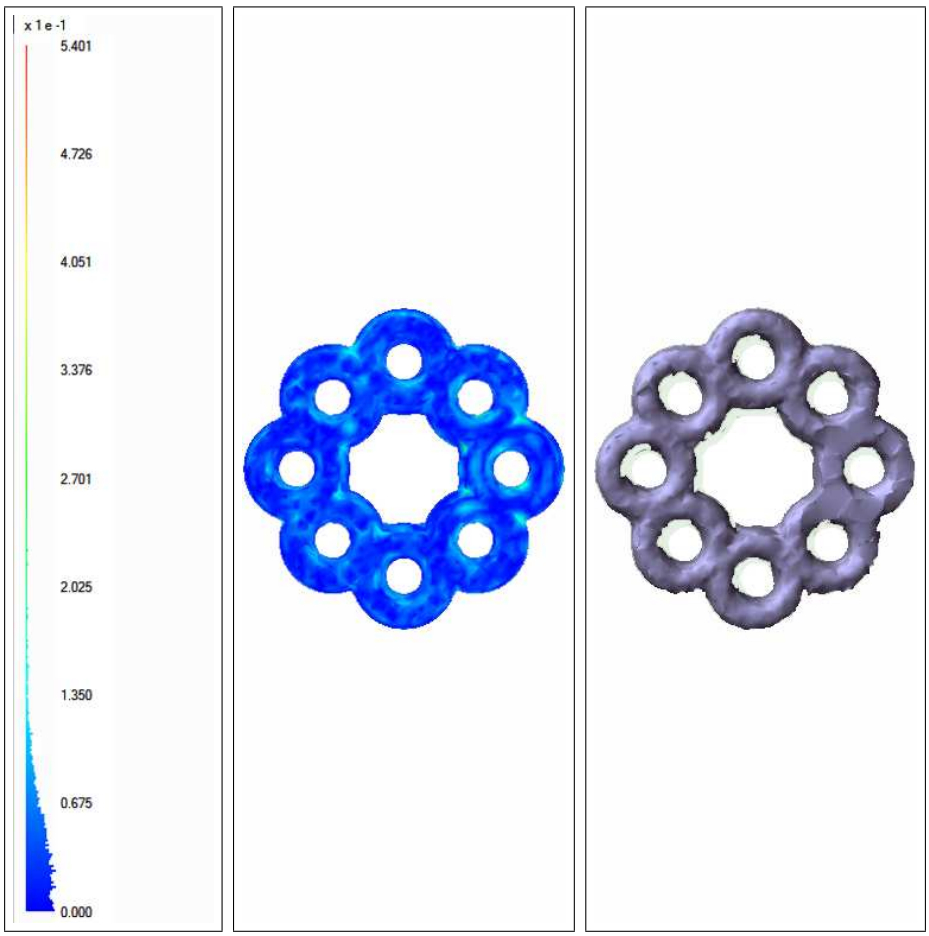
(a) (b) (c)

Figure 3.21: 9 Handle Torus model compressed with JPEG2000 to 14.6 KB (12.4 bpv).



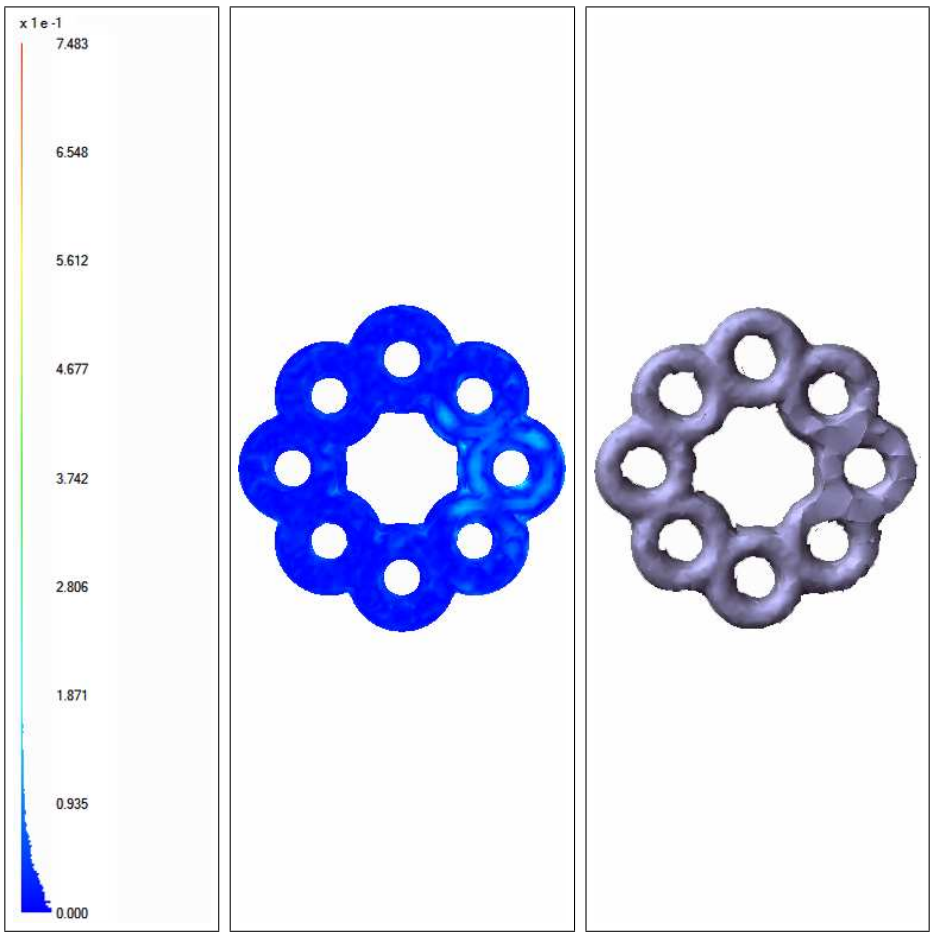
(a) (b) (c)

Figure 3.22: 9 Handle Torus model compressed with JPEG2000 to 13.6 KB (11.6 bpv).



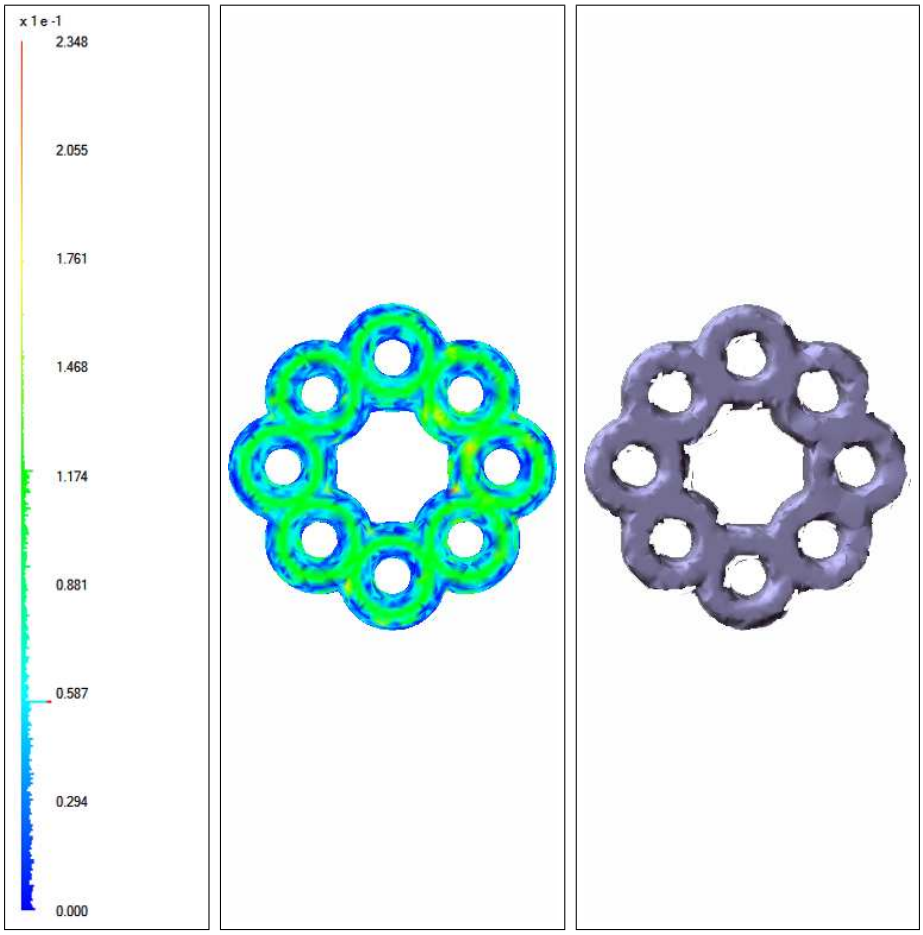
(a) (b) (c)

Figure 3.23: 9 Handle Torus model compressed with JPEG2000 to 14 KB (11.9 bpv).



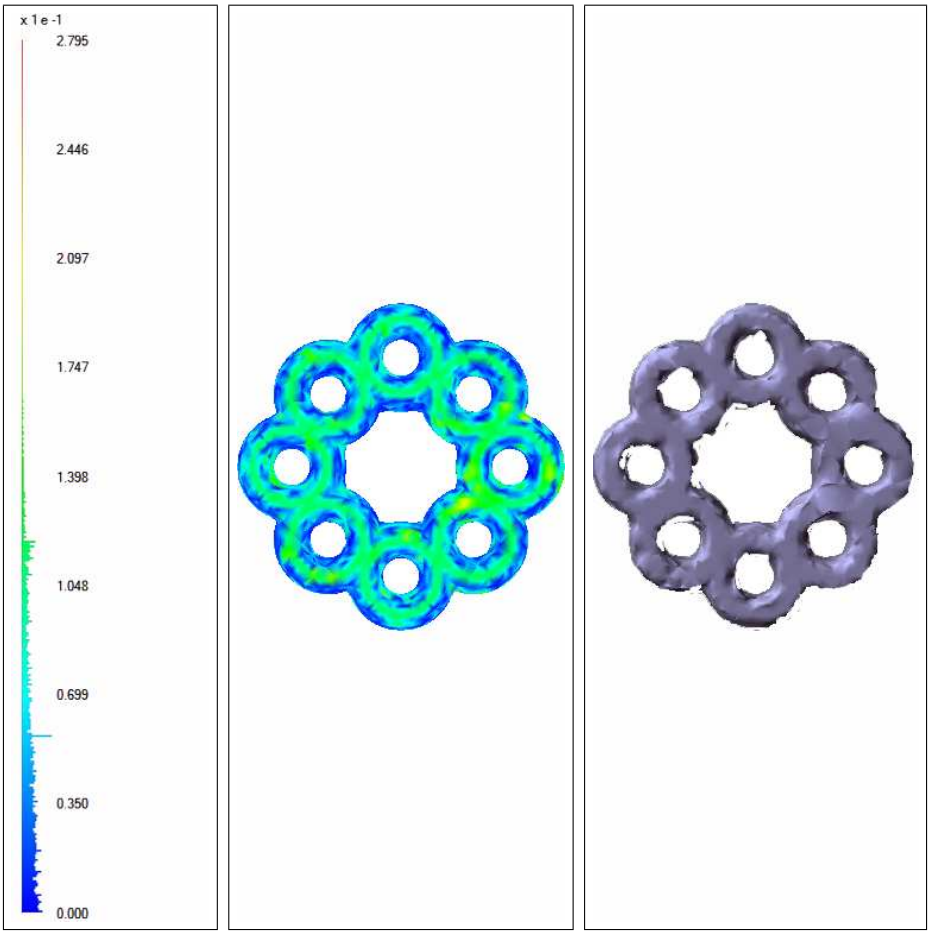
(a) (b) (c)

Figure 3.24: 9 Handle Torus model compressed with JPEG2000 to 16.7 KB (14.2 bpv).



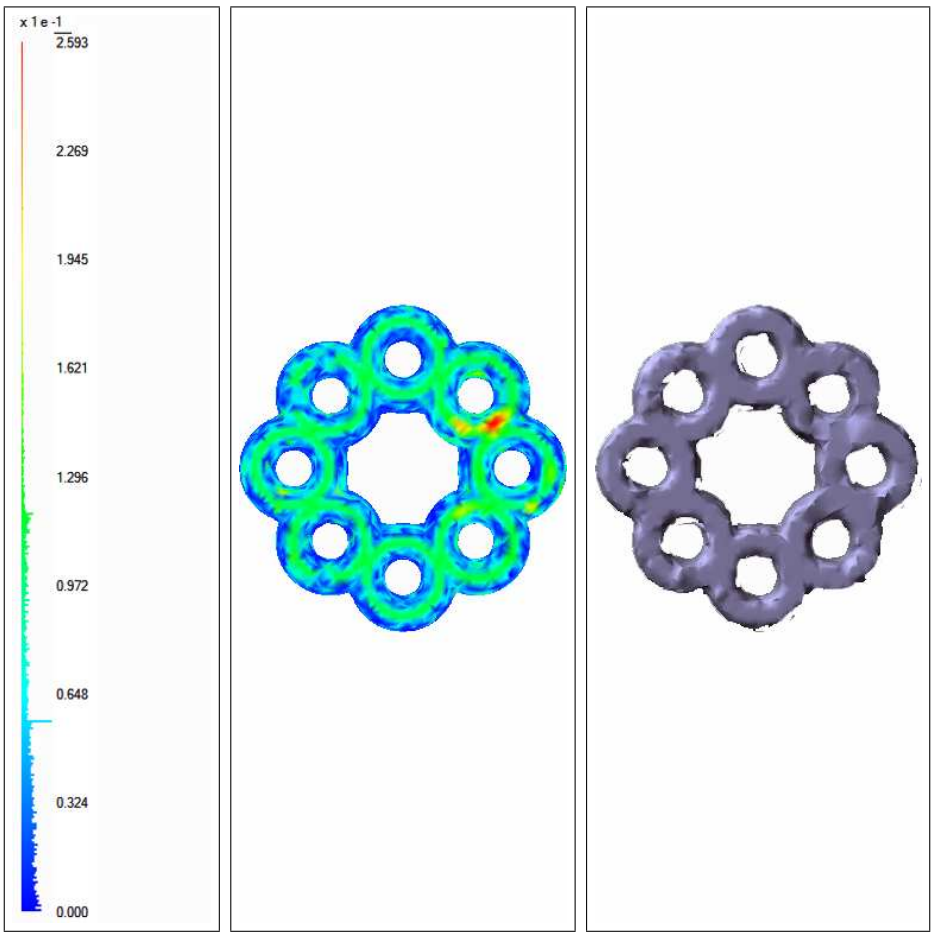
(a) (b) (c)

Figure 3.25: 9 Handle Torus model compressed with SPIHT to 7.84 KB (6.7 bpv).



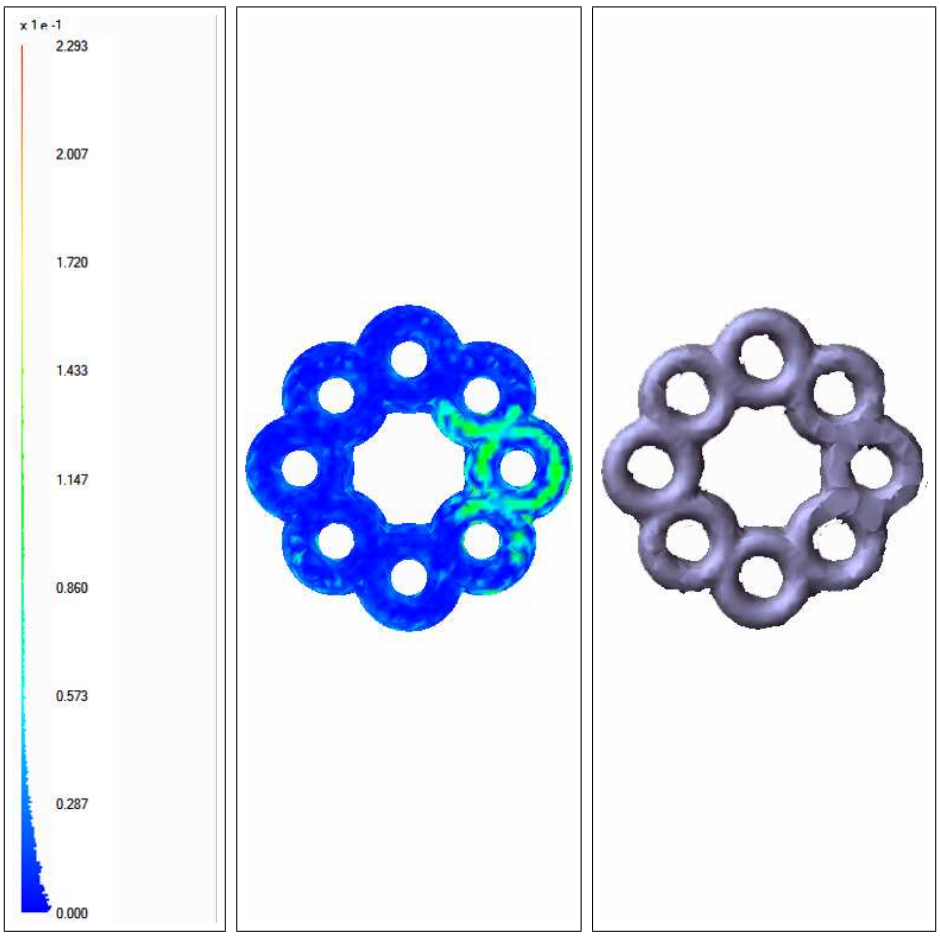
(a) (b) (c)

Figure 3.26: 9 Handle Torus model compressed with SPIHT to 8.18 KB (7 bpv).



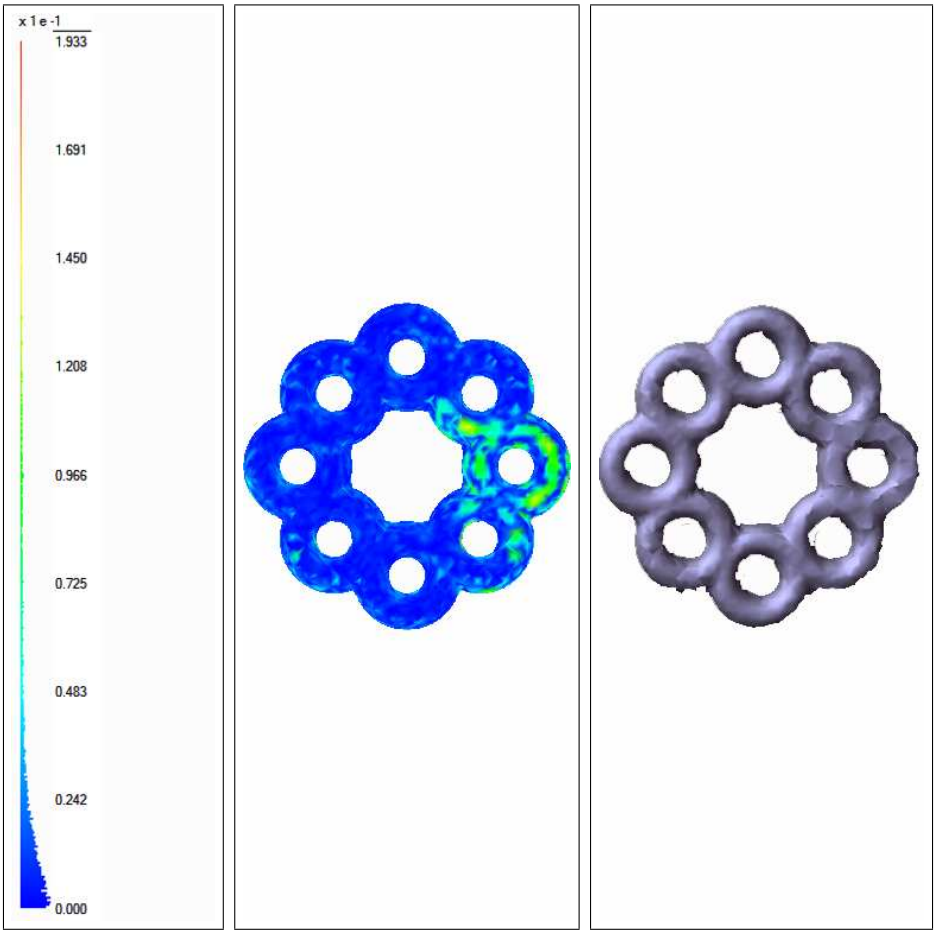
(a) (b) (c)

Figure 3.27: 9 Handle Torus model compressed with SPIHT to 8960 KB (7.63 bpv).



(a) (b) (c)

Figure 3.28: 9 Handle Torus model compressed with SPIHT to 11.9 KB (10.1 bpv).



(a) (b) (c)

Figure 3.29: 9 Handle Torus model compressed with SPIHT to 12.7 KB (10.8 bpv).

Chapter 4

Conclusions and Future Work

In this thesis, a new mesh compression framework that uses connectivity-guided adaptive wavelet transform based image coders is introduced. Two newly defined concepts are : projection and the connectivity based wavelet transform. Here in this thesis it is shown that a mesh can be compressed using ready-made wavelet based image compression tools. Also it is shown that by using these tools both single rate and progressive encoding can be achieved.

The results in this thesis show that the idea of using image processing tools on meshes can be realized without making any parameterizations [5] on the mesh or manipulations on the used image processing tools [7]. Furthermore, the results given in Chapter 3 show that the use of newly defined connectivity-guided adaptive wavelet increases the encoding efficiency against the use ordinary wavelet transform.

The projection operation defined here is simple to implement and needs less computation than the parameterization introduced in [5]. The parameterization approach needs the mesh to be cut and opened homeomorphic to a disc. Then by solving several linear equations, mesh is transformed to an image. Despite

parameterization, projection operation needs only two computationally easy linear equations to be solved: one for determining the pixel position of the mesh vertex on the image and one for the finding the values of the projected pixels.

Although the introduced approach is simple it has some drawbacks. Since the 3D mesh models are not homeomorphic to a disk, most of the time it is not possible to find correspondence between each vertex in a mesh and an image pixel. So some of the vertices get lost in the during the projection operation. To handle those situations, a neighborhood based interpolation scheme which predicts the values of the lost vertices from its projected neighbors is defined.

However, it is observed that if many vertices are lost, this algorithm does not work well. Especially for complex models this can be the situation. So for complex models, more than one projection of the mesh should be taken. This increases the bit rates but decreases the distortion level of the reconstructed mesh. A way to find the best projection, by which maximum number of mesh vertices are projected, must found to improve the distortion rates.

Using projection operation an image-like representation of the 3D mesh is created. The new image-like structure can be encoded using any image coding tool. Here it is shown that the idea of using wavelet transform gives the opportunity of creating a progressive representation for the mesh. Since the correlation between the pixels in the image-like representation is different from the ordinary images, an algorithm like the defined connectivity-guided adaptive wavelet transform is needed to better exploit the correlations.

In Tables 3.7 - 3.8 with results in [1], it can be said that for the same distortion level, adaptive scheme has lower bit rates than non-adaptive scheme. Also Figure 3.15 visually shows that for nearly the same rate, SPIHT coder has lower distortion than the JPEG2000 coder. Thus, adaptive approach is superior to the

non-adaptive approach. This is due to the better exploitation of the correlation between connected vertex positions.

The used connectivity-guided adaptive wavelet transform is based on the prediction of grid points from its neighbors. Due to this reason, both SPIHT and JPEG2000 encoders provide better results when the values of signal samples are more highly correlated with each other. Hence, a mesh-to-image transform providing high correlation between the grid values will lead to higher compression ratios.

The best compression algorithms in literature use remeshing as the preliminary step, since they are not applicable to irregular meshes. Much smaller errors reside while making predictions on the vertices of those models. Thus, their compression bit rates are very low compared to the compression bit rates of algorithms that are compressing irregular meshes. But they can not reconstruct the original mesh since they remeshed the model. Also the wavelet based compression algorithms like PGC in [44] in literature are only applicable to remeshed models. The advantage of our algorithm is that it is applicable to any mesh model structure like; irregular, semi-regular or regular.

The future work will include finding the best projection plane so that maximum number of vertices are projected on the image-like representation and more correlation between the neighboring pixels arise. Adaptive approach can be applied to the image-like representation using other wavelet bases. It is known that due to their big support other wavelet bases do not give good results in a non-adaptive manner. However adaptiveness may change this situation since it redefines the neighboring concept in every subband of image-like representations.

The future work will also include the compression of the dynamic meshes using video compression methods. In dynamic meshes, there exists a inter-vertex correlation between vertices of mesh sequence. Instead of compressing

each mesh frame separately, group of mesh frames should be coded together. Thus the inter-vertex correlation between mesh frames can be exploited. Using video compression algorithms are suitable for purpose.

Bibliography

[1] **K. Köse, A. E. Çetin, U. Güdükbay, L. Onural, "Nonrectangular wavelets for multiresolution mesh analysis and compression", *Proc. of SPIE Defense and Security Symposium, Independent Component Analyses, Wavelets, Unsupervised Smart Sensors, and Neural Networks IV*, Vol. 6247, pp. 19-30 2006.**

[2] **M. Lounsbery, T. D. DeRose, J. Warren, "Multiresolution Analysis for Surfaces of Arbitrary Topological Type", *ACM Transactions on Graphics*, Vol. 16-1, pp. 34-73, 1997.**

[3] **M. Deering, "Geometry Compression", *Proc. of ACM SIGGRAPH*, pp. 13-20, 1995.**

[4] **C. E. Shannon "A mathematical theory of communication", *Bell Syst. Tech. Journal*, Vol. 27, pp. 379-423, 623-656 1948.**

[5] **X. Gu, S. Gortler, H.Hoppe, "Geometry Images", *Proc. of ACM SIGGRAPH* pp.355-361, 2002.**

[6] **A. Sheffer, E. Praun, K. Rose, Mesh Parameterization Methods and their Applications, *Foundations and Trends in Computer Graphics and Vision*, *Publishers Inc.* 2006.**

[7] I. Guskov, W. Sweldens and P. Schröder, "Multiresolution signal processing for meshes", *Proc. of ACM SIGGRAPH*, pp. 325-334, 1999.

[8] J. Peng, C.-S. Kim, Kuo, and C.-C. Jay, "Technologies for 3D triangular mesh compression: a survey", *Journal of Visual Communication and Image Representation*,Vol. 16, No. 6 , pp. 688-733, 2005.

[9] S. Gumhold, Mesh Compression, PhD Dissertation, 2000

[10] Weisstein, W. Eric, "Homeomorphic" From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/Homeomorphic.html

[11] J. Rossignac, "Edgebreaker: connectivity compression for triangle meshes", *IEEE Trans. on Visualization and Computer Graphics*, Vol. 5, No. 1, pp. 47-61, 1999.

[12] M. Mantyla, "An Introduction to Solid Modeling", *Computer Science Press* 1988.

[13] Bruce G. Baumgart, "A Polyhedron representation for computer vision", *Proc. of the National Computer Conference*, pp. 589596, 1975.

[14] K.Weiler, "Edge-based data structures for solid modeling in curved-surface environments", *IEEE Computer Graphics and Applications* Vol. 5, No. 1, pp. 21-40, 1985.

[15] R. C. Gonzalez, R. E. Woods, Digital Image Processing, *Prentice Hall*, 2002.

[16] P. Alliez, M. Desbrun, "Valence-driven connectivity encoding of 3D meshes", *Computer Graphics Forum*, Vol. 20, pp. 480-489, 2001.

[17] F. Móran and N. Garcia, "Comparison of wavelet-based three-dimensional model coding techniques", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 14, No. 7, pp. 937-949, 2004.

[18] G. Taubin, J. Rossignac, "Geometry compression through topological surgery" *ACM Transactions on Graphics*, Vol. 17, No. 2, pp. 84-115 1998.

[19] C. Touman, C. Gotsman, "Triangle mesh compression" *Proc. Graphics Interface*, pp. 26-34, 1998.

[20] Z. Karni and C. Gotsman, "Spectral compression of mesh geometry", *Proc. of ACM SIGGRAPH*, pp. 279-286, 2000.

[21] H. Hoppe, "Progressive meshes", *Proc. of ACM SIGGRAPH*, pp. 99-108, 1996.

[22] E. Catmull and J. Clark, "Recursively generated B-spline surfaces on arbitrary topological meshes", *Computer Aided Design*, Vol. 10, pp. 350-355, 1978.

[23] I. Guskov, K. Vidimce, W. Sweldens, P. Schröder, "Normal meshes". *Proc. of ACM SIGGRAPH*, 2000.

[24] R. Ansari and C. Guillemort, "Exact reconstruction filter banks using diamond FIR filters", in *Proc. Bilcon*, pp. 1412-1424, 1990.

[25] L. Ibarria, J. Rossignac, "*Dynapack*:space-time compression of the 3D animations of triangular meshes with fixed connectivity", Technical Report, TR-No. 03-08, 2003.

[26] J. Zhang, C. B. Owen, "Octree-based Animated Geometry Compression" *Data Compression Conference*, pp. 508-517, 2004.

[27] K. Muller, A. Smolic, M. Kautzner, P. Eisert, T. Wiegand, "Predictive compression of dynamic 3D Meshes", *Proceedings of International Conference on Image Processing*, pp. 621-624, 2005

[28] J. Zhang, C. B. Owen, "Hybrid Coding for Animated Polygonal Meshes: Combining Delta and Octree", *International Conference on Information Technology: Coding and Computing*, Vol. 1, pp. 68-73, 2005

[29] J. Rossignac, A. Szymczak, "*Wrap and Zip*: Linear decoding of planar triangle graphs", *IEEE Trans. Visual. Computer Graphics*, Vol. 5, No.1, pp. 47-61, 1999

[30] M. Isenburg, J. Snoeyink, "Compressing the property mapping of polygon meshes", *Proc. of Pacific Graphics*, pp. 4-11, October 2001.

[31] E. Lee and H. Ko, "Vertex data compression for triangular meshes", *In Proc. of Pacific Graphics*, pp. 225-234, 2000.

[32] O. Sorkine, D. Cohen-Or, and S. Toldeo, "High-pass quantization for mesh encoding", *In Proc. of Eurographics Symposium on Geometry Processing*, pp. 42–51, 2003.

[33] R. Cohen, D. Cohen-Or and T. Ironi, "Multi-way Geometry Encoding", *Technical report* ,2002.

[34] C. Loop, "Smooth subdivision surfaces based on triangles", *Master Thesis-University of Utah*, 1987.

[35] J. Ho, K-C. Lee, D. Kriegman. "Compressing Large Polygonal Models", *Proc. of IEEE Visualization Conference Proc.*, pp.357-362, 2001.

[36] M. Isenburg, S. Gumhold, "Out-of-Core Compression for Gigantic Polygon Meshes", *Proc. of ACM SIGGRAPH*, pp. 935-942, 2003.

[37] P. Alliez, C. Gotsman, "Recent advances in compression of 3D meshes", *In Proc of Symposium on Multiresolution in Geometric Modeling*, 2003.

[38] A. Szymczak, J. Rossignac, D. King, "*Piecewise Regular Meshes*: Construction and Compression", *Graphics Models (Special Issue on Processing of Large Polygonal Meshes)*, 2003.

[39] J. Popovic, H. Hoppe, "Progressive simplicial complexes", *Computer Graphics*, Vol. 31, No. Annual Conference Series, pp. 217-224 1997.

[40] G. Taubin,A. Gueziec, W. Horn, F. Lazarus, "Progressive forest split compression", *Computer Graphics*, Vol. 32, pp.123-132, 1998.

[41] P. Alliez, M. Desbrun, "Progressive compression for lossless transmission of triangle meshes", *Proc. of ACM SIGGRAPH*, pp. 198-205, 2001.

[42] P. Schröder, W. Sweldens, "Digital geometry processing", *Course Notes, ACM SIGGRAPH*, 2001.

[43] A. Khodakovsky, P. Schröder, W. Sweldens, "Progressive geometry compression", *Proceedings of the* 27^{th} *annual conference on Computer Graphics and interactive techniques*, pp 271-278, 2000

[44] A. Khodakovsky, I. Guskov, "Normal mesh compression", *Proc. of the 27th Annual Conference on Computer Graphics and interactive techniques*, pp. 95-102, 2000.

[45] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, D. Dobkin. "MAPS: Multiresolution adaptive parameterization of surfaces", *Computer Graphics, Annual Conference Series*, Vol.32, pp.95104, 1998.

[46] C. Loop, *Smooth subdivision surfaces based on triangles*, Master's Thesis, University of Utah, Department of Mathematics, 1987.

[47] N. Dyn, D. Levin, J. A. Gregory, "A butterfly subdivision scheme for surface interpolation with tension control", *ACM Transactions on Graphics*, Vol. 9-2 pp. 160-169, 1990.

[48] M. Garland and P. Heckbert, "Surface simplification using quadric error metrics", *Proc. of ACM SIGGRAPH*, pp. 209-216, 1997.

[49] A. Said and W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees, *IEEE. Trans. Circ. Syst. Video Tech.*, Vol. 6, pp. 243–250, 1996.

[50] Ö. N. Gerek, A.E. Çetin, "Adaptive polyphase subband decomposition structures for image compression" *IEEE Transactions on Image Processing*, Vol. 9 No. 10 pp. 1649-1660, 2000

[51] J.M. Shapiro, "Embedded image coding using zerotrees of wavelets coefficients", *IEEE Trans. Signal Processing*, Vol. 41, pp. 3445-3462, 1993

[52] JPEG 2000 Part I Final Committee Draft, Version 1.0, 2000.

[53] http://mathworld.wolfram.com/Projection.html.

[54] C. Guillemot, A.E. Çetin, and R. Ansari "M-channel nonrectangular wavelet representation for 2-D signals: basis for quincunx sampled signals" *Proc. of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 4, pp. 2813-2816, 1991.

[55] S. Arivazhagan, D. Gnanadurai, J.R. Antony Vance, K.M. Sarojini, and L. Ganesan. "Evaluation of zero tree wavelet coders", *Proc. of International Conference on Information Technology: Computers and Communications*, p. 507, 2003.

[56] C. Valens, EZW encoding, Available at http://perso.wanadoo.fr/polyvalens/clemens/ezw/ezw.html.

[57] S. Mallat, A Wavelet Tour of Signal Processing, *Academic Press.*, 1999.

[58] G. Strang, T. Nguyen Wavelets and Filter Banks *Wellesley Cambridge*, 1996.

[59] M. D. Adams "The JPEG-2000 Still Image Compression Standard".

[60] D. Novosel, M.Kovač "JPEG2000 software implementation" *Proc. of* 46^{th} *International Symposium Electronics in Marine - ELMAR-2004*, pp. 573-578, 2004

[61] J.-L. Gailly, gzip Compression Software.

[62] J. Ziv and A. Lempel, "A universal algorithm for data compression", *IEEE Trans. on Information Theory*, Vol. 23, No. 3, pp. 337-343, 1977.

[63] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: measuring error on simplified surfaces", *Computer Graphics Forum*, Vol. 17, No. 2, pp. 167-174, 1998.

[64] S. Gumhold, W. Strasser, "Real-Time compression of triangular mesh connectivity", *Proc. of ACM SIGGRAPH*, pp 133-140, 1998.

[65] M. Isenburg, P. Alliez "Compressing Polygon Mesh Geometry with Parallelogram Prediction" *IEEE Visualization*, pp. 141-146, 2002.

[66] R. Pajarola and J. Rossignac, "Compressed progressive meshes", *IEEE Trans. on Visualization and Computer Graphics*, Vol. 6, No. 1, pp. 79-93, 2000.

[67] H. Hoppe, E. Praun, "Shape compression using spherical geometry images", *N. Dodgson, M. Floater, M. Sabin (Eds.), Advances in Multiresolution for Geometric Modelling*, Springer-Verlag, pp. 2746, 2005.

[68] G. Rote, "Computing the Minimum Hausdorff Distance Between Two Point Sets on a Line Under Translation", *Information Processing Letters*, Vol. 38, No. 3, pp. 123-127, 1991.

[69] N. Aspert, D. Santa-Cruz and T. Ebrahimi,"MESH: Measuring Error between Surfaces using the Hausdorff distance", *Proc. of the IEEE International Conference on Multimedia and Expo (ICME)*, Vol. 1, pp. 705-708, 2002