ISTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

VIEW DEPENDENT CODING OF 3D SCENES

M.Sc. THESIS

Muzaffer Ege ALPER

Department of Computer Engineering

Computer Engineering Programme

DECEMBER 2011

ISTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

VIEW DEPENDENT CODING OF 3D SCENES

M.Sc. THESIS

Muzaffer Ege ALPER
(504081521)

Department of Computer Engineering

Computer Engineering Programme

Thesis Supervisor: Assoc. Prof. Dr. Uluğ BAYAZIT

DECEMBER 2011

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

3 BOYUTLU SAHNELERİN GÖRÜNTÜ TABANLI KODLANMASI

YÜKSEK LİSANS TEZİ

Muzaffer Ege ALPER
(504081521)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Assoc. Prof. Dr. Uluğ BAYAZIT

ARALIK 2011

Muzaffer Ege ALPER, a M.Sc. student of ITU Institute of Science and Technology 504081521 successfully defended the thesis entitled **"VIEW DEPENDENT CODING OF 3D SCENES"**, which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. Uluğ BAYAZIT** Istanbul Technical University	.....
Jury Members :	**Prof.Dr. Muhittin Gökmen**	.....
	Assoc.Prof.Yücel Yemez	.....
	Assoc.Prof.Uluğ Bayazıt	.....

Date of Submission : **19 December 2011**
Date of Defense : **26 January 2012**

FOREWORD

I thank my supervisor, Assoc. Prof. Uluğ BAYAZIT, for introducing me to the theory of probability and its applications. Through his guidance, I was able to learn a big deal on different aspects of Applied Probability in my Masters study. I would also like to thank all of the workers and professors, whose collective labor constitutes the University and create a stimulating environment to study and research.

December 2011

Muzaffer Ege ALPER
Computer Engineer

TABLE OF CONTENTS

Page

ABBREVIATIONS

RD	**:** Rate Distortion
SSD&N	**:** Screen Space and Normal Distortion
PSNR	**:** Peak Signal-to-Noise Ratio
CSPECK	**:** Color Spectral Transform Coder
MAPS	**:** Multiresolution Adaptive Parameterization of Surfaces
MSE	**:** Mean Squared Error

LIST OF TABLES

LIST OF FIGURES

Page

VIEW DEPENDENT CODING OF 3D SCENES

SUMMARY

3 dimensional graphics technology allows one to examine a visual object in many angles. Thus, naturally, many compression methods are optimized without considering the viewpoint and direction of the observer. However, there are many applications in which one can imagine objects in a scene are only examined from particular viewpoints. Background objects and objects that move synchronously with the camera are examples of this scenerio. In this case there is a lot more room for compression, due to not only excluding the invisible parts from the coding process but also the psycho-visual properties of the objects.

There have been several methods proposed to exploit these properties and compress the meshes even further. However, these methods are either optimal in very strict conditions or heuristic in nature. Thus, a general purpose method, that would be optimal under various conditions, is lacking. Consequently, the main question to be answered in this Thesis work is "is there a method that would optimize the compression directly for the final image while still being computationally practical ?".

The road to the answer contains discussion and comparison of the previously proposed methods and a novel rendering based method that is optimal for the final image. This optimality comes at a high cost in terms of computational resources. The balance is achieved by a hybrid method that initially computes an approximate solution in a very fast manner and then refines this solution via the slower but more optimal method. We show that such a method significantly improves the viewpoint dependent compression performance, surpassing the proposed methods in the literature, while being fast enough for practical applications.

3 BOYUTLU SAHNELERİN GÖRÜNTÜ TABANLI KODLANMASI

ÖZET

Günümüzde bilgisayar grafik teknolojileri giderek yaygınlaşmaktadıtır. Bu teknolojiler üretimden, eğlenceye ve sağlığa kadar farklı alanlarda uygulama olanağı bulmuşlardır. Ancak giderek çoğalan veri sayısı, bu verilerin etkin kodlanmasını zorunlu kılmaktadır. Bilgisayar grafiği teknolojileri arasında çokgen tabanlı, 3 boyutlu yüzey modelleri, matematiksel basitlikleri ve temsil kolaylığı gibi nedenlerle en sık kullanılan araçlardandır. 3 Boyutlu bilgisayar grafikleri teknolojileri sayesinde bir nesnenin birden fazla bakış açısı altında görülebilmesi mümkün olmuştur.

Herhangi bir veri sıkıştırma yöntemi, verideki istatistiksel ilişkileri ve izleyicinin görsel algılama özelliklerini değerlendirmelidir. Günümüzde veri sıkıştırmanın istatistiksel doğası oldukça iyi anlaşılmış durumdadır ve bu bilgi yeni ve daha güçlü veri sıkıştırma yöntemleri için kullanılmaktadır. Ancak izleyicilerin görsel algılama özelliklerine dayalı çalışmalar nispeten daha seyrektir ve karmaşıklık kaygıları nedeniyle daha az uygulama bulmaktadır. Ancak kesin sonuçlar yerine, yaklaşıklıklar kullanıldığında, bu karmaşıklık sorununu gidermek mümkündür.

3 boyutlu nesneleri temsil etmek için çeşitli yöntemler geliştirilmiştir. Bunlar genel olarak 2 grupta incelenebilir: hacim tabanlı (voxel) olanlar ve yüzey tabanlı olanlar. Yüzey tabanlı nesne temsili, opak nesnelerin sadece dışının görüleceği, dolayısıyla sadece nesnenin dışının çizilmesinin yeterli olduğu fikrine dayanır. Yarı-opak nesneler de çeşitli yaklaşımlar ile bu şekilde modellenebilir. Bu temsil biçimi, özellikle eğlence, eğitim ve prototip geliştirme uygulamalarında yaygın olarak kullanılmaktadır. Çokgen tabanlı modeller arasında, üçgen modeller, basitlikleri ve düzlemselliği garanti etmeleri sebebi ile en sık kullanılır. Bu çalışmada da üçgen tabanlı yüzey modelleri esas alınmıştır.

En genel haliyle, 3 boyutlu bir görüntü, herhangi bir bakış noktasından incelenebilir. Bu nedenle, doğal olarak, çoğu veri sıkıştırma metodu bakış açısından bağımsız olarak kaliteyi değerlendirerek eniyileme yapar. Bu en iyileme için genellikle, 3 boyutlu uzaydaki karesel hata kullanılır. Karesel hata basit bir bozunum ölçütüdür, ancak buna rağmen görülen bozunumu belirli bir başarıyla yansıtır. Bu çalışmada da bazı yerlerde karesel hata ölçütünden yararlanılmıştır.

Bir nesnenin belirli bakış noktalarından görülebileceği durumlar da düşünülebilir. Arkaplandaki nesneler ve kamera ile aynı hız ve doğrultuda hareket eden nesneler buna örnektir. Bu nesnelerin 3 boyutlu kodlanması belirli bakış noktalarından izlenmekten çok (bunun için doğrudan 2 boyutlu resmi de kullanılabilir), farklı aydınlatma koşullarında ve farklı nesnelerle ilişki içinde görselleştirmeye hizmet etmektedir. Bu durumda sıkıştırma yönteminde bakış noktasına uygun eniyileme yapılabilir. Örneğin

uzakta olan nesneler daha kaba bir şekilde kodlanırken, yakındaki nesneler daha ayrıntılı ele alınarak basit bir “hız ataması” sağlanabilir.

3 Boyutlu bir sahnede, tel file modelleri sıkıştırmadan önce ele alınması gereken bazı temel ön işlemler vardır. Bunlardan en önemlisi, sahnede görünmeyen yüzlerin ve düğümlerin atılmasıdır. Bu işlem sayesinde, bit bütçesi sadece görünen yüzeyler için harcanmış olacaktır. Görünmeyen yüzlerin atılması için, bu Tez'de, hızlı bir ışın-üçgen kesişimi algoritması kullanılmıştır. Burada her bir düğüme, izleyicinin bakış noktasından bir ışın gönderilir. Bu ışının kestiği en yakın üçgenin, ilgili düğüme komşu olmaması durumunda, o düğüm görünmez olarak işaretlenir. Bir yüzün (çokgen), tüm köşelerinin görünmez olması durumunda o yüz sahneden atılır. Her bir kesişim işleminin hızlı olması, işlem sayısının düğüm sayısına karesel olarak bağlı olması nedeniyle, toplam işlemin kısa olmasını gerektirmez. Özgün tel file modeller üzerinde çalışan görünmeyen yüzlerin atılması işleminin oldukça pahalı olması nedeniyle, bu Tez'de tel file modelin uygun bir parametrikleştirilmesi yoluyla işlemi hızlandıran yeni bir yöntem önerilmektedir. Burada, açı koruyan bir parametrikleştirme işlemi yardımıyla, ilgili tel file model, daha basit bir tel file model üzerinde parametrik olarak temsil edilir. Böylece bu basit model üzerinde yapılan görünmeyen yüzlerin atılması işlemi sonuçları tekrar (ters fonksiyon aracılığıyla) özgün modele taşınır. Açı koruyan parametrikleştirme işlemi sayesinde, %95'e varan hız kazanımları gözlemlenmiştir. . Işlemin yaklaşık bir sonuç vermesi nedeniyle oluşan hatanın ise küçük olduğu gözlemlenmiştir. Önerilen yöntem, başarım eğrisinde farklı süre-başarım noktaları seçilmesine olanak vermektedir, böylece ihtiyaca göre süre ve başarım performansı ayarlanabilir.

Birçok nesnenin bulunduğu bir sahnenin sıkıştırılması sırasında gerekli bir başka ön işlem ise sahnenin bölgelere ayrılmasıdır. Bu işlem kodlayıcı karmaşıklığını düşürür ve video kodlamadaki bloklara ayırma işlemine benzetilebilir. Bu işlemle birlikte hangi bölgenin, toplam bit bütçesinden ne kadar bit kullanacağı sorusu hız ataması probleminin daha genel bir hali olarak ortaya çıkar. . Izleyici konumuna bağlı kodlamada ise bu çok daha önemli bir sorun olur, çünkü bu durumda sahnedeki bölgeler arasındaki algılanabilirlik farkları büyümüştür. Literatürde önerilen çeşitli yöntemler bu sorunsala farklı çözümler getirir. Ancak bu yöntemler ya kısıtlı koşullar altında optimaldir, ya da sezgisel yöntemlerdir ve hangi koşullarda optimal olduklarından bahsedilmez. Farklı görselleştirme koşullarında eniyi sonucu verecek genel bir yöntem eksiktir. Bu nedenle, bu Tez'de şu temel soruya cevap aranmaktadır: "hem çok çeşitli koşullarda eniyi olan hem de kaynak kullanımı açısından gerçekçi sınırlarda olan bir yöntem mümkün müdür?".

Literatürde daha önce önerilen ve bu çalışmada incelenen yöntemler arasında, “ekran uzayı”ndaki bozunumları eniyileyen, en fazla görünen kısımların 3 Boyutlu karesel hatalarını eniyileyen, izleyiciye en yakın olan kısımların 3 Boyutlu karesel hatalarını eniyileyen yöntemler vardır. Tez içinde, bu yöntemlerin hız ve başarımları karşılaştırılmış, üstün ve eksik yanları gösterilmiştir. Yukarıda belirtilen sorunun cevabına varmak için, öncelikle literatürdeki yöntemler ile bu çalışmada ilk olarak önerilen ve hesaplama karmaşıklığı yüksek imge tabanlı yöntem karşılaştırılmıştır. Literatürdeki bazı yöntemlerin oldukça hızlı olabildiği gözlemlenirken, önerilen yöntem ile arada başarım farkının bulunduğu gösterilmiştir.

Bu gözlemler üzerine, hesaplama karmaşıklığı açısından diğer yöntemlerle rekabet edebilecek ama başarım açısından imge tabanlı yönteme yakın bir yöntem tasarlanmıştır. Bu yeni yöntem için en temel gözlem, uzaklık tabanlı hız atama yönteminin oldukça düşük bir hesaplama karmaşıklığı ve yüksek çalışma hızı ile görüntü tabanlı yöntemin sonucuna oldukça yaklaşabilmesidir. Uzaklık tabanlı hız atama yönteminin vardığı çözüm başlangıç noktası kabul edilerek, adım adım, görüntü tabanlı bozunum ölçütüne göre daha iyi sonuçlar elde eden karma bir yöntem önerilmektedir. Yapılan deneyler sonucunda önerilen yöntemin, literatürdeki diğer yöntemlerden üstün olduğu gözlemlenmiştir. Ekte ise yöntemin hangi koşullar altında eniyi sonucu vereceği kuramsal olarak tartışılmıştır.

1. INTRODUCTION

With the increasing amount of computational resources available for 3D graphics, the use of computer graphics technologies is becoming more common. Today even handheld devices are capable of processing and displaying detailed virtual 3D environments. The standard PC configurations are capable of displaying huge and highly detailed environments in real time. The ever increasing penetration of 3D graphics in many aspects of our lives results in an increasing demand on developing better and more efficient 3D processing methods for compression in various applications.

As 3D mesh compression (geometry, connectivity) methods matured, researchers began to divert from generic compression and pursue more subtle issues such as joint texture-mesh coding, scalability and so on [1, 2, 3]. One such topic is the view-dependent rate allocation for different objects/regions in a given scene. For reasons related to complexity, many compression methods process input meshes by partitioning them into regions and handling the regions individually. A drawback is a loss in fidelity of reconstruction due to ignoring statistical dependencies among regions. However, one can turn this into an advantage by employing rate-allocation to different regions. Such a system can optimize based on a possibly complex distortion function, which is not necessarily the same function for which the specific coder of 3D data of each region is optimal. In our case, the use of a rate-distortion optimization procedure allows us to optimize the coded scene in terms of visual quality, something the coder is unaware of.

The problem of rate distortion optimized view dependent coding of 3D scenes/objects has been addressed previously in [4, 5, 6]. Under the assumption of a fixed viewpoint, a primary goal is to allocate rate from a limited bit budget among different coding units (i.e. regions in a 3D Object) so that the overall distortion is minimized. Possible

applications can vary from single resolution compression of scenes from a fixed viewpoint to progressive transmission of scenes from time-varying viewpoints.

The rate allocation methods developed so far can be categorized into heuristic (distance based [6] and visibility based [4] methods) and analytical methods. It is difficult to talk about the optimality of the heuristic methods but these methods tend to have a wider applicability since they use more 'general' features of the meshes, in the sense that they are not directly related to a specific illumination method or another specific condition. On the other hand, methods like illumination based and/or screen space distortion based optimization [5] are optimal for a specific rendering model (such as Phong shading model). However, the optimality is limited to these criteria and rendering parameters(for example, the particular shading method used) and optimality for other rendering parameters is not discussed in [5].

One of the problems of these rate-allocation methods for view-dependent coding/transmission of scenes is the lack of a performance bound which would indicate how much room there is to improve the current methods. This difficulty is in part due to the lack of a measure for objective assessment of the visual quality of a scene. Perceptive models (such as those mentioned in [7] and [8]) that consider the perceptibility of features to the human visual system, constitute an important advance in this area. These models were not considered in this Thesis.

The main objective of this thesis is to propose an image based rate-allocation method that minimizes the squared error in the rendered image. This method is also used as a practical upper bound for the other methods. Practically, however, the image based method lags behind [4, 5, 6] in terms of computational complexity. To remedy this situation; a hybrid method combining the image based method with the simple method of [6] is proposed and shown to be both practically feasible and have a better performance than previously proposed alternatives in the literature.

View-dependent coding methods typically need to remove invisible faces of the scene before the actual rate-allocation computations begin [6]. However, complexity of a brute force procedure, that determines and culls all invisible faces, may hinder it from being employed for large meshes. In this work, we propose an alternative method

based on mesh simplification and parameterization, that may be preferable to space partitioning methods in certain situations.

Another contribution of this thesis work is a unified discussion and comparison of the various view dependent rate-allocation methods that were previously proposed in the literature.

This thesis is organized as follows:

- In Chapter 2, the coding methods used in the Thesis are introduced.
- Chapter 3 states the problem of rate-allocation and compares the rate-allocation performances of "implicit" rate allocation performed by the coder and the optimal method.
- Chapter 4 discusses a required step in view-dependent compression: removal of the invisible parts of the seen. The exact method based on ray/triangle intersection is found to be slow and a novel, faster approximate method is introduced.
- Chapter 5 is the core of this Thesis. First several methods previously proposed for view dependent rate allocation are discussed, together with the newly considered image based method that yields optimal quality in the final rendered image. The section, then, continues with the proposed hybrid method that is both practically feasable and very close to being optimal.
- Finally, Chapter 6 concludes the Thesis and indicates directions for further improvements.

2. CODING METHODS

The goal of any lossless visual signal compression method is to reduce both the statistical and psycho-visual redundancies [9]. While the possibility of compressing a signal without loss exists, such methods are limited in their ability to reduce the stream size. Ideally, a signal $\vec{x}$ can be represented in $-log_2(p(\vec{x}))$ bits, where this amount is known as the self information. Thus we can hope, at best, for an average coding rate of $E(-log_2(p(\vec{x})))$ bits. The quantity $E(-log_2(p(\vec{x})))$ is known as the entropy of the source.

It's the problem of lossy compression where things get more interesting. Similar to the lossless case, the reduction of size is due to statistical properties of the signal (the rate distortion function) [10] and psycho-visual properties. The rate distortion function $R(D)$ indicates the infimum of the possible amount of bits to spend in order to code the source with at most D distortion. While the theory is not constructive, it has proved useful to the signal compression community as a lower bound, so that they can assess their methods better.

The Digital Geometry Processing (a.k.a. Graphical Signal Processing) community, however, do not have the same opportunity. While there are such attempts [11], the 3D graphical models (meshes) have been too complicated to allow statistical modeling. Thus we have to work without a reassuring lower bound.

In this chapter, we review the 3D mesh compression methods that are used later in the thesis. The discussion includes an outline of the methods (together with suggested improvements, if any) and the intuitions behind them.

2.1 Spectral Transform and CSPECK Coding

The algorithm used to code 3D meshes in [12] is based on the spectral transform [13] (analogous to the DCT applied on 1D or 2D regular topologies) which uses a matrix

operator analogous to 2D 2nd derivative operator (the eigenvectors of this 2D derivative operator yields discrete cosine transform vectors and eigenvalues are related to the frequencies). This operator is called the Laplacian matrix and is defined as:

$$L_{ij} = \begin{cases} 1 & \text{i=j} \\ -1/d_i & \text{i adjacent to j} \\ 0 & \text{else} \end{cases} \quad (\mathbf{2.1})$$

Here d_i shows the degree of vertex i. The eigenvectors of this matrix form an orthogonal basis in R^n, similar to discrete cosine basis in DCT, and the eigenvalues correspond to "frequencies". To get a better understanding of the significance of Laplacian matrix for meshes we refer the reader to [11]. For each coordinate x,y and z, the spectra are coefficients computed by projecting the respective coordinate vector onto the eigenvectors.

After the spectra coefficients are computed, they are coded by 3D CSPECK method adapted from the 2D CSPECK proposed for transform coding of color images in [14]. This method works by coding bits of spectra coefficients in decreasing order of their significance using a recursive set partitioning method.

To code the coefficients we make use of the concept of significance of a set T_i, a set of coefficients corresponding to the ith axis, which is defined for a particular value of n, the "pass", as

$$\Pi_n(T_i) = \begin{cases} 1 & 2^n \leq \max_{j \in T_i} |c_{i,j}| < 2^{n+1} \\ 0 & \text{else} \end{cases} \quad (\mathbf{2.2})$$

Here $c(i, j)$ is the jth component of coordinate vector belonging to ith axis.

In the algorithm, we make use of two types of sets: S and I. In the first pass, the coefficient vectors are partitioned into sets of I and S and added to the list of insignificant sets (LIS). At each pass, a significance test is performed on each set in the LIS. Significant sets are removed from LIS and are further partitioned until the significant coefficients are located. These coefficients are added to the list of significant points (LSP) and the sets not deemed significant in the process are added to LIS. After this, a refinement step takes place where the less significant bits of the coefficients added to LSP before the current pass are output. Finally, the significance level n is reduced by one in the quantization step and the next pass is started. This procedure is repeated until the desired rate is reached. The partitioning procedures

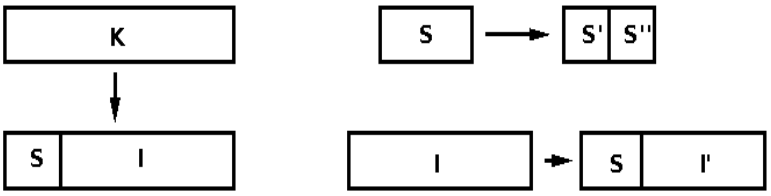

Figure 2.1: Set partitioning.

for different sets are illustrated in Figure 2.1. This method makes use of the fact that coefficients with larger magnitudes (constituting most of the energy) are concentrated around lower frequencies while high frequency coefficients generally have smaller magnitudes. This way, many zeros (corresponding to insignificant coefficients) can be coded with a single insignificance decision. The decisions made during the coding such as set significance, coefficient significance and refinement bits are entropy coded using an arithmetic coder.

The original algorithm codes coefficients of different regions and coordinates in an interleaved manner. This results in an implicit rate allocation among different regions and coordinates. This allocation, being suboptimal, is nevertheless good [12]. Another benefit of such an interleaved coding is embeddedness and progressive transmission capability. Embeddedness refers to the capability of using a single output file to decode at different rates (which are smaller than the actual coding rate), while progressive transmission refers to the case where data is ordered in its decreasing content of information.

In order to have a finer control of rate allocation among regions, we can choose to handle each region separately and solve the rate allocation problem explicitly as explained in [15]. The coded regions are then output sequentially. Such a coding, however, means that the bitstream no longer possesses embeddedness or progressive transmission capability, since bits corresponding to smaller significance level tests of a region precede those corresponding to greater significance levels of other regions.

2.2 Valence-Based Connectivity Coder

Connectivity is another source of information, necessary to completely specify a triangular mesh. In its simplest form, connectivity can be encoded by binary

neighbourhood matrices. A component of this matrix is 1 if the two vertices indicated by the row and column are connected by an edge. However, such a representation is very wasteful since the valence distribution of the vertices of natural models are contrained. Space requirements for this scheme is obviously $O(v^2)$ bits, where v is the number of vertices, so we need v bits per vertex.

Another representation is explicitly defining (triangular) faces of the mesh. This representation lists the id values of the corner vertices for each face. It turns out that, statistically, this is not the optimal coding scheme either. Here the space requirement is $O(3 * f * \log_2 v)$, for a planar triangular graph with no boundaries we get $f = 2 * v - 4$ [16]. Combining these formulas, we obtain a rate of almost $6 * \log_2 v$ bits per vertex.

In [17], Gotsman showed that, if we assume all possible planar graphs with v vertices have the same probability (i.e. come from a uniform distribution), we have an entropy value of 3.24bpv. It means that, we can do much better than a logarithmic bit rate and constant average bit rate is possible. Note that, this entropy value includes some pathological triangulations, which are very unlikely to occur in practice. Since this entropy value is calculated under the principle of maximum entropy, we can get much better average coding rates for more natural distributions.

In [16], a single resolution method that achieves rates in range of 0 to 4 bpv is proposed. The main observation leading to the method is that all "natural" meshes have a distribution of valences that has a mean and a peak close to 6 and falling gradually away from it. Thus we can expect a good compression performance if we can code connectivity by using valences, since the entropy of the valences will be small. In [17], it is shown that the entropy of valences in a single (planar) mesh is upper bounded by 3.23 bpv. Since this is smaller than 3.24, we need other information in order to be able to code any planar triangulation, aside from valence values. The proposed method, achieves this by additional "split" and "merge" operations, together with valence values. By keeping the number of extra operations as low as possible, the coder achieves very good compression rates. In [17], average coding rate of 1.8 bpv is reported for "natural" meshes. Now we shall discuss the method and related operations in more detail.

2.2.1 Preprocessing: hole patching

The connectivity coding algorithm expects 2-manifold orientable meshes without boundaries as input. Thus any non-2manifold mesh must be first cut into 2-manifold submeshes and any holes must be patched using a dummy vertex and dummy faces. Cutting meshes to satisfy 2manifold property will not be discussed in this thesis (i.e. we assume 2-manifold meshes as input) but a mechanism to patch holes is discussed.

Our method uses a block decomposition approach to find the individual holes. The resulting holes are then patched by adding a single dummy vertex in the middle and dummy faces connecting this vertex with the vertices on the boundary of the hole. From now on, we shall mention the boundary edges and vertices (the boundary graph), simply as the graph. The block decomposition algorithm use a depthfirst search tree to find cutvertices of the graph and then use these to find the blocks.

First we need to find the cut-vertices of the graph. The following pseudo-code illustrates the algorithm.

Data: a connected graph G
Result: a set K of cut-vertices of the graph G
Initialize the set K as empty ;
Choose an arbitrary vertex r of G ;
Execute a depthfirst search of G, starting at r ;
Let T be the output dftree ;
if *root r has more than one child in T* **then**
    Add r to K;
end
foreach *vertex w* **do**
    compute low(w);
end
foreach *nonroot v* **do**
    if *there's a child w of v such that low(w)* $\geq$ *dfnumber (v)* **then**
        Add v to set K.;
    end
end
return K;

Algorithm 1: Finding cut vertices.

In the above algorithm, dfnumber(v) shows the iteration of df search algorithm in which the vertex v is first seen. low(v) is the smallest of 1)dfnumber(v) , 2)

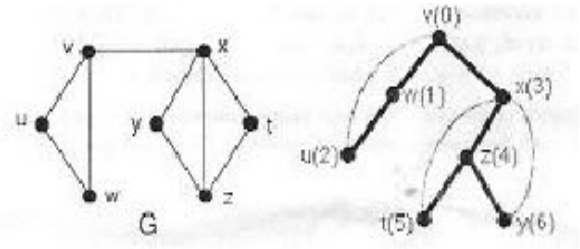

Figure 2.2: A sample graph and its df tree with dfnumbers in parenthesis.

dfnumber(z) for any z joined to v by a nontree edge or 3) low(y) for any child y of v.

Figure 2.2, shows a sample graph, together with the dfnumbers of vertices.

The idea behind the algorithm can be briefly expressed like this: If any descendant w of a vertex v is not connected with a nontree edge to any vertex in the graph which has a lower dfnumber than v (i.e. An ancestor of v) , then any path from any vertex outside the subtree rooted at w, to w must pass through v. Thus, v is a cut vertex.

From Figure 2.2 we can say that any path starting from u,v or w to y,z or t must pass through x, so x is a cut vertex and indeed low(z) = 3 $\geq$ 3 = dfnumber(x).

After the computation of the cutvertices, the blocks of a graph can be found using this algorithm:

Data: a connected graph G
Result: the vertex sets B_i of the blocks of the graph G
Find the cutvertices of G;
Initialize the block counter i = 0;
foreach *cutvertex v of G (in order of decreasing dfnumber)* **do**
    foreach *child w of v in df tree T* **do**
        if *low (w)* $\geq$ *dfnumber(v)* **then**
            Let Tw be the subtree of T rooted at w;
            i++;
            B_i = vertices of Tw $\cup$ v;
            T = T vertices of Tw ;
        end
    end
end
return sets B_i;

Algorithm 2: Decomposing a graph into its blocks

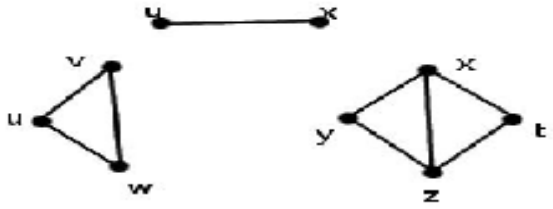

Figure 2.3: Decomposition of the graph.

In Figure 2.3, we see a sample graph and the resulting output after decomposition.

Now we can state the overall hole patching algorithm.

Data: a 2-manifold graph with boundary
Result: a 2-manifold graph without boundary
Initialize the dummy counter i = 0;
Add all edges of graph G which have only 1 incident face to set L;
3: Select any vertex v in L;
Using this vertex as root find blocks in the component of L containing v;
foreach *block B* **do**
    add a dummy vertex di to G and connect it to all vertices in B;
    remove B from L;
end
if *L is empty* **then**
    return G;
end
else
    go to 3;
end

Algorithm 3: Hole patching algorithm.

2.2.2 Valence based connectivity coder

In this section, we describe the valence based connectivity coding method [18]. We shall assume that our input is properly processed and is a 2-manifold mesh without boundaries.

First we need to define some important concepts that will be used to express the algorithm.

Conquest edge list (Active List) : A set of edges forming a simple cycle. All vertices and edges in the list have "active" status. The algorithm operates on elements in the active list, and during the course of the "conquest", the list can be expanded,

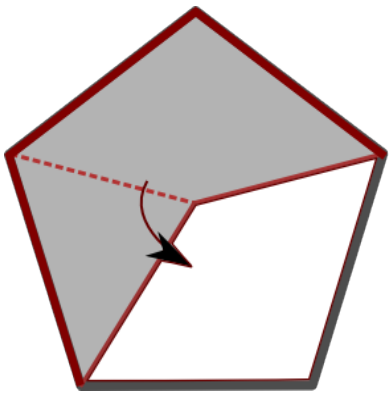

Figure 2.4: Edge conquest. Active list is highlighted in red.

shrank, split and merged. This list locally separates the surface into "inner" and "outer" regions. Finally the edges in this list are directed, so that for each vertex in the active list, there's exactly one incoming and one outgoing edge. Consistency of these directions is maintained in the conquest, split or merge operations.

Pivot (Focus) Vertex: A vertex chosen for conquering its adjacent vertices in counterclockwise fashion.

Free vertex: Default state of a vertex. A free vertex has not been encountered by the conquest yet.

Conquered edge: An edge whose both adjacent faces have been coded.

Conquered vertex: A vertex with every edge conquered.

Dummy vertex: A vertex that is artificially added to the mesh to remove one hole.

The conquest is initialized by selecting an initial triangular face, adding it to the active list and selecting a focus vertex in that face. It should be noted that edges in the active list have directions. A focus is processed by selecting its next "free" adjacent vertex (in the counter clockwise order as seen on Figure 2.4), outputting its valence, setting it active and adding it to the active list. The edge arriving to the focus in the active list is flagged as conquered and removed from the active list. The process is shown in Figure 2.4.

During an edge conquest one of four different cases will occur:

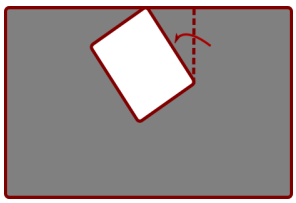

Figure 2.5: Split. Active list is highlighted in red.

1. The target vertex is free. It's now flagged as active, its valence is output and its added to the active list.

2. The target vertex is a dummy vertex. It's tagged as active and a dummy code followed by valence is output. It's also added to the list appropriately.

3. The target vertex is already active and in this active list. Then a split code is output followed by the order of the vertex in the active list. The active list is split into two and one of them is pushed to the active list stack. This step is necessary to resolve the local ambiguity.

4. The target vertex is active but belongs to another active list. We then output a merge code followed by the order of the vertex in the list. Finally the necessary data to code the neighbouring vertices on one side of the merged vertex is output and the lists are merged.

The way our implementation handles merges is a bit different from the original method [18]. Merging of lists may cause a vertex to have not one but two incoming active edges which cause ambiguity. To resolve this issue, we output additional bits to identify the particular edge when traversing these vertices. While this means outputting more bits than the amount specified in the original papers, the merge event is so rare (actually it can only occur if genus is greater than 0) that the resulting coding rates are virtually the same.

Following two figures (2.5,2.6), illustrate the split and merge events.

After all neighbouring vertices are processed we conquer one last edge for free and tag the focus as conquered and remove the vertex from the active list. We then choose

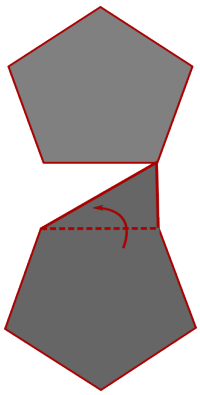

Figure 2.6: Merge. Active list is highlighted in red.

another vertex as focus. These operations are repeated until all active lists have been fully processed, henceforth the mesh is coded.

One last point about the algorithm is on the selection of new focus vertices. The original method chose a new focus vertex by a deterministic process. That is the new focus is the next active vertex in the active list. P. Alliez and M.Desbrun [16] noticed that this resulted in many split codes which are expensive. They proposed an improvement on the selection of focuses by using an adaptive method. For each vertex in the active list a score using weighted sum of the remaining free edges of the the vertex and of its neighbouring active vertices is computed. The vertex having the lowest score is chosen as the next focus. In our implementation we have used equal weights to compute the scores. Such a selection dramatically reduces the number of split codes.

Table 2.1: Connectivity Coding Results.

Mesh	Num. of Vertices	Num. of Splits	Rate (per vertex)
Triceratops	2832	0	1.86
Bunny	34834	0	1.98
Eight	766	4	0.34
Femur	14988	11	2.26
Venus	50002	12	2.38

Another difference in our implementation is that we choose new focus vertices after each edge conquest (before all free neighbours are conquered) but in the original papers the next focus is chosen only after the current one is fully conquered (i.e. No remaining free edges incident to it). We have observed that our method reduces the number of splits even more, leading to slightly higher compression efficiency.

Once a code is determined it's fed to an adaptive arithmetic coder. This coder uses the frequencies of the input symbols to efficiently compress them. On average the resulting bit stream's size is very close to the entropy of the valence distribution of the graph. Table 2.1 outlines the results.

3. RATE-ALLOCATION PROBLEM

In this chapter, we introduce the problem of rate-allocation in a formal way. For the time being, the discussion will assume MSE distortion in the 3D space. In the consequent chapters, we shall see how the discussed methods are also appropriate in a viewpoint dependent coding setting.

The bit (rate) allocation process aims to find a distribution of a limited number of bits among multiple quantizers such that the overall distortion is minimized. Let B be the set of all possible bit allocation vectors. For n quantizers and distortion measure $D(.)$, we can formally state the problem as:

$$\underline{b}^* = \text{argmin}_{\underline{b} \in B} D(b) \tag{3.1}$$

subject to the constraint,

$$R(\underline{b}) = \frac{1}{v} \sum_{i=0}^{n} b_i \leq R_c \tag{3.2}$$

where $\underline{b} = (b_1, b_2, ..., b_n)$, b_i is the number of bits allocated to quantizer i, $D(\underline{b})$ is the overall distortion for the allocation $\underline{b}$, $\underline{b}^*$ is the optimal allocation and v is the number of coefficients(in our case, vertices) to be coded.

First of all, we categorize the rate allocation methods based on whether the optimization is explicitly computed using an optimization algorithm (explicit rate allocation) or is a side product of the coding system (i.e. "computed" implicitly - Implicit rate allocation).

3.1 Explicit Rate Allocation

The methods which solve the explicit rate allocation problem can be categorized into i)model based (parametric) and ii)non-model based (nonparametric) methods. The most widely used and successful mesh compression methods are transform based methods (spectral transform, wavelet transform). Model based rate allocation methods model the transform coefficients by choosing a distribution among a family

of distributions and analytically solving for the optimal allocation [19], using the rate-distortion function of the distribution. Model based methods are generally much faster than non-model based methods. Model based methods use MSE as the error metric, since they lead to easier analytical solutions.

Nonparametric rate allocation methods take another approach. Instead of assuming a probabilistic model for the coefficients, they collect rate-distortion points (called operating points) using the actual coder, thereby forming a discrete operational rate-distortion function. The methods then employ a particular optimization method to find the optimal operating point ($\underline{b}^*$). The advantages of this approach include avoidance of statistical mismatches between the background distribution of natural 3D models and the assumed distribution and the capability of being employed with any coding method *and distortion measure*. An important drawback is that the distortion measurements for various rates must be collected before the actual coding starts which may be a time consuming operation.

There are multiple methods that can be used to compute the optimal allocations, given the operational rate-distortion function. One approach is dynamic programming [20]. The results obtained by dynamic programming methods are precise, but the process has high complexity. The more popular approach to solve the non-parametric rate allocation problem is based on Lagrangian multipliers based optimization [21, 15]. These methods have the advantage of being fast but are also less precise (i.e. they provide exact solutions only at points which lie on the convex hull of the operational rate distortion function, otherwise they fall short of the intended rate value).

In this Thesis, we make use of the nonparametric rate allocation method based on Lagrangian multipliers proposed by Shoham and Gersho [15] to solve the rate allocation problem. This method works well even if the quantizers have different characteristics (such as having non-convex quantizer functions). We show below the main theorem for the algorithm.

Rate Allocation Theorem: For any $\lambda \geq 0$ the solution $\underline{b}^*(\lambda)$ for the unconstrained problem,

$$\underline{b}^*(\lambda) = \text{argmin}_{\underline{b} \in B}\{D(\underline{b}) + \lambda R(\underline{b})\} \qquad \textbf{(3.3)}$$

is also the solution of the problem (**3.1**) with the constraint

$$R_c = R(\underline{b}^*(\lambda)) \tag{3.4}$$

With this knowledge, we can find optimal allocations for various rate constraints by sweeping λ from a high value (which would result in the allocation of least possible number of bits to all quantizers) to a small value and thereby determine the λ value achieving a target rate. There are actually smarter methods to find the λ value corresponding to a given rate. For a detailed account of faster techniques for determining λ refer to [15, 21].

When the quantizers are operated independently of each other, the minimization problem of (**3.3**) can be solved for each quantizer(coding unit) separately. In the current context, each quantizer governs the coding process of each region in the scene. We solve for each region $b_i^* = argmin_{b_i}(D_i(b_i) + \lambda b_i)$ for some λ that satisfies $\frac{1}{v}\sum_i b_i(\lambda) = R(\lambda) \leq R_c$. Since, for a particular solution, the same λ value is used for all quantizers, this solution is also known as the "constant slope solution". Figure 3.1 shows this allocation on a simple two quantizer problem.

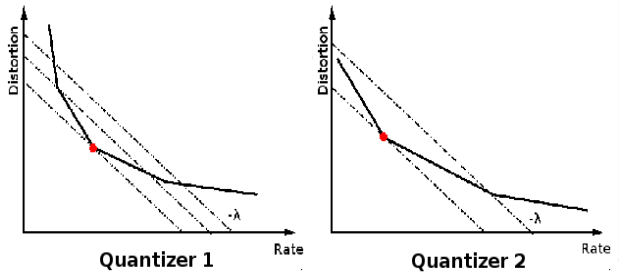

Figure 3.1: Rate-Distortion optimization. The optimal operating points, indicated as red dots, for all quantizers have the same slope.

This method is equivalent to recursively assigning rate to quantizers, such that the assignment (the marginal rate and the region to which this rate is added) with the maximum marginal gain ($\frac{\Delta D}{\Delta R}$) is selected at each step until the desired rate is met (or no more possible assignment has a positive marginal gain) [15]. This is how the method is implemented.

3.2 Implicit Rate Allocation

The coding system we use is based on coding spectral coefficients [13] of the scene using CSPECK method [14, 12]. CSPECK algorithm, as described before, makes "passes" on the regions in the scene, outputting coefficient information until the desired rate is reached. Such a system implicitly provides a rate allocation, based on minimizing the maximum possible distortion (i.e. the distortion metric for the scene is, $\max_i D_{MSE}(b_i)$, the maximum MSE distortion among the regions). In this section, we shall first briefly describe how we code multiple regions together and then compare the results to explicit rate allocation using MSE.

3.2.1 Coding multiple objects

Define $\mathscr{M}$ as the set of points of the mesh M . The regions $\mathscr{P}_i$ formed by the partitioning algorithm satisfies $\mathscr{P}_i \subset \mathscr{M}, \bigcup_{i=0}^{n} \mathscr{P}_i = \mathscr{M}$ and $\mathscr{P}_j \bigcap \mathscr{P}_i = \emptyset$ for $i \neq j$ where n is assumed to be the number of regions. The original method formed an array $\{a_i\}$ of regions and processed all of these regions in each pass in order. To code multiple meshes in the same session we modify these definitions slightly: Let $\mathscr{M}_j$ be the set of points of the mesh M_j, $\mathscr{P}_{ij}$ be the ith region of the jth mesh. We now form an array $\{a_i\}$ such that if $a_n = \mathscr{P}_{i_0 j_0}$ and $a_m = \mathscr{P}_{i_1 j_1}$, where if $n > m$, then $i_0 \geq i_1$. In other words we do not code the $(i+1)$th region until the ith region of all meshes (whose total number of regions $n_j \geq i$) are coded. Consider the following case as an example: We have two meshes M_0 and M_1 whose regions are $(\mathscr{P}_{00}, \mathscr{P}_{10})$ and $(\mathscr{P}_{01}, \mathscr{P}_{11}, \mathscr{P}_{21}, \mathscr{P}_{31})$ respectively. The array that defines the order of processing is like this : $\{\mathscr{P}_{00}, \mathscr{P}_{01}, \mathscr{P}_{10}, \mathscr{P}_{11}, \mathscr{P}_{21}, \mathscr{P}_{31}\}$. The reason we order the regions like this is to get a more balanced distribution of bits (and consequently the distortions) among all objects.

3.2.2 Results

The distortion metric for these examples is the MSE metric, as shown below;

$$\sum_{v \in \mathscr{M}} (1/n) ||\underline{v} - \underline{v}'||^2 \tag{3.5}$$

Table 3.1: Rate Distortion data for Beethoven and triceratops models.

λ	Rate	$D_e(M)$	$D_i(M)$	$D_i(M_2)$	$D_i(M_1)$	$D_e(M_2)$	$D_e(M_1)$
$0.9 \cdot e-5$	23.97	1.91e-05	2.13e-05	2.13e-05	2.13e-05	2.01e-05	1.80e-05
$1 \cdot e-5$	23.67	2.05e-05	2.28e-05	2.29e-05	2.27e-05	2.33e-05	1.75e-05
$1.5 \cdot e-5$	22.89	2.6e-05	2.79e-05	2.74e-05	2.83e-05	3.34e-05	1.80e-05
$2 \cdot e-5$	21.23	4.28e-05	4.65e-05	4.70e-05	4.60e-05	3.35e-05	5.27e-05
$2.5 \cdot e-5$	19.17	9.24e-05	9.70e-05	9.76e-05	9.64e-05	1.29e-04	5.34e-05
$3.5 \cdot e-5$	17.31	1.64e-04	1.80e-04	1.82e-04	1.79e-04	1.29e-04	2.02e-04
$4.5 \cdot e-5$	16.38	2.26e-04	2.43e-04	2.42e-04	2.43e-04	2.50e-04	1.99e-04
$5 \cdot e-5$	15.43	3.41e-04	3.45e-04	3.45e-04	3.44e-04	4.72e-04	2.02e-04
$6 \cdot e-5$	14.15	4.9e-04	5.23e-04	5.17e-04	5.29e-04	4.72e-04	5.08e-04
$6.3 \cdot e-5$	13.68	5.72e-04	6.26e-04	6.20e-04	6.32e-04	4.72e-04	6.79e-04
$6.8 \cdot e-5$	13.54	6.01e-04	6.65e-04	6.69e-04	6.60e-04	4.72e-04	7.40e-04
$7.6 \cdot e-5$	11.76	1.05e-03	1.09e-03	1.12e-03	1.06e-03	1.34e-03	7.40e-04
$10 \cdot e-5$	9.84	1.81e-03	1.99e-03	1.95e-03	2.03e-03	1.78e-03	1.84e-03

where $\mathscr{M}$ is the set of all reconstructed points in the scene and $\underline{v}$ is the original values of the reconstructed verices $\underline{v}'$ and n is the number of points.

For the first test, two low resolution meshes are used: Beethoven (M_1): 2655 points, 5028 faces; triceratops (M_2): 2832 points, 5660 faces. These meshes are of comparable sizes.

The scene (consisting of both of these meshes) is referred to as M (i.e. $M = M_1 \cup M_2$). $D_i(.), D_e(.)$ are distortion functions for the implicit and explicit rate allocation cases. The results can be seen in Table 3.1.

Table 3.2: Rate Distortion data for venus and Horse models.

λ	Rate	$D_e(M)$	$D_i(M)$	$D_i(M_1)$	$D_i(M_2)$	$D_e(M_1)$	$D_e(M_2)$
$0.3 \cdot e-8$	19.01	8.50e-09	1.57e-08	1.63e-08	1.40e-08	9.26e-09	7.73e-09
$0.4 \cdot e-8$	17.81	1.24e-08	2.51e-08	2.30e-08	3.02e-08	1.67e-08	8.04e-09
$0.6 \cdot e-8$	16.36	1.89e-08	4.09e-08	4.00e-08	4.29e-08	1.67e-08	2.10e-08
$0.9 \cdot e-8$	16.16	2.06e-08	4.36e-08	4.39e-08	4.29e-08	1.94e-08	2.18e-08
$1 \cdot e-8$	15.41	2.74e-08	5.41e-08	5.65e-08	4.81e-08	2.57e-08	2.91e-08
$2 \cdot e-8$	12.75	6.41e-08	1.26e-07	1.26e-07	1.26e-07	6.37e-08	6.45e-08
$3 \cdot e-8$	12.70	6.56e-08	1.27e-07	1.27e-07	1.29e-07	6.66e-08	6.45e-08
$4 \cdot e-8$	11.80	9.75e-08	1.70e-07	1.79e-07	1.47e-07	1.30e-07	6.45e-08
$5 \cdot e-8$	11.75	9.99e-08	1.73e-07	1.84e-07	1.47e-07	1.35e-07	6.45e-08
$6 \cdot e-8$	10.95	1.47e-07	2.26e-07	2.51e-07	1.64e-07	2.31e-07	6.45e-08
$10 \cdot e-8$	0.90	1.50e-07	2.32e-07	2.59e-07	1.64e-07	2.37e-07	6.45e-08
$20 \cdot e-8$	9.25	3.58e-07	4.2e-07	4.89e-07	2.45e-07	6.52e-07	6.45e-08

Table 3.3: Rate Distortion data for venus and Horse scaled.

λ	Rate	$D_e(M)$	$D_i(M)$	$D_i(M_1)$	$D_i(M_2)$	$D_e(M_1)$	$D_e(M_2)$
$0.7 \cdot e-7$	17.51	2.12e-07	2.14e-07	2.19e-07	2.01e-07	2.31e-07	1.63e-07
$1 \cdot e-7$	16.76	2.52e-07	2.84e-07	2.93e-07	2.62e-07	2.31e-07	3.06e-07
$2 \cdot e-7$	14.76	5.20e-07	5.86e-07	6.23e-07	4.95e-07	4.90e-07	5.96e-07
$4 \cdot e-7$	12.41	1.07e-06	1.37e-06	1.36e-06	1.39e-06	8.62e-07	1.61e-06
$8 \cdot e-7$	10.75	2.12e-06	2.14e-06	2.19e-06	1.95e-06	2.35e-06	1.61e-06
$10 \cdot e-7$	10.26	2.35e-06	2.38e-06	2.34e-06	2.36e-06	2.43e-06	2.25e-06

In the second test two other meshes are used, venus (M_1): 50002 points, 100000 faces and Horse (M_2): 19851 points, 39698 faces. One important property of this configuration is that the starting indices of the meshes in the bitplane are different (For this case the difference is 3 levels). As we see in Table 3.2 this will result in a worse implicit rate allocation.

In our experiments we have been able to pin down two main issues that partially explains this behaviour. The first cause is the different impact of different bitplane indices of the meshes on the distortion. This is best explained by an example. In our test case of venus and Horse, the starting bitplane index of venus is 3 and of Horse is 0. This difference occurs due to the fact that our transform is orthonormal and causes the energy to concentrate on the lower eigenvalues. Suppose CSPECK mesh coder is running its third pass on the scene, this means it is coding the 3rd index of venus but the 1st index of Horse. The current implementation gives both of these the same importance but actually the affect of the most significant bits of bitplane of the Horse model has a bigger impact on the overall distortion than the third most significant bits of that of venus. That is due to the fact that the coder has to spend more rate to code the bitplane in the 3rd index of venus than the first index of horse. The results in Table 3.2 also confirm this as we can see the optimal allocation has smaller distortion (more rate) for the Horse model.

The second issue is the bits used for the smaller mesh before the coder even begins to actually code the coefficients. This happens because of the difference of the respective starting indices of the meshes, for instance at the first pass of the same test case when the most significant bits of the venus model's coefficients are coded, the zeros of the other model also has to be coded. Considering that this coding happens for all

partitions in the mesh, we can see that this waste has an impact (even though a very small one) on the overall rate-distortion performance since these bits could have been used to actually code coefficients, this also has a negative impact on the arithmetic coders' probability tables. We can see this affect on the first row of Table 3.2 where the individual distortions of *both* the meshes are bigger when coded together even though the same total number of bits are used as the optimal case. This could be solved by sending not only the largest starting index but the starting indices of all of the coded objects to the decoder. For this, of course, additional bits should be transfered to the decoder to identify the regions of different objects.

To verify these arguments, Table 3.3 shows data where the same meshes (venus and Horse) are used except that they were scaled beforehand so that their initial coefficients are almost equal.

It should be noted, however, that the only comparison between the explicit and implicit rate allocation schemes is not their rate-distorion behaviour but also their different complexities. We have seen that on a Ubuntu system running on a Pentium 4 3.0 GHz machine, explicit rate allocation computations of the venus-Horse test case was almost 3 times slower than implicit allocation. Even though the numbers are not exact since the tests were not optimised, this shows the less optimal behaviour of the CSPECK mesh coder in a scene with varying sizes of objects might still be preferable in some cases.

4. PREPROCESSING - INVISIBLE FACE CULLING

In this chapter, invisible face culling operation, as a preprocessing step, is described. The discussion of this operation is divided into two subsections for ease of understanding: i) the naive method, ii) the proposed approximate method.

4.1 Invisible Face Culling

4.1.1 The basic method

In view-dependent coding, we strive to find a rate-allocation solution such that rate is expended in proportion to the contribution of the individual parts to the the final quality of the scene. Thus, assuming a fixed viewpoint scenario, it makes sense to remove the unseen parts of the scene, since they do not contribute at all. Similar to the process explained in [6], the unseen faces are removed from the scene using a fast ray/triangle intersection method [22].

To explain the operation better, we need some definitions. We define, $\vec{edge}1$, $\vec{edge}2$, $\vec{pvec}$ ($\vec{pvec} = \vec{edge}1 \times \vec{edge}2$), $\vec{tvec}$ and $\vec{dir}$ as in Figures 4.1, 4.2, 4.3, 4.4.

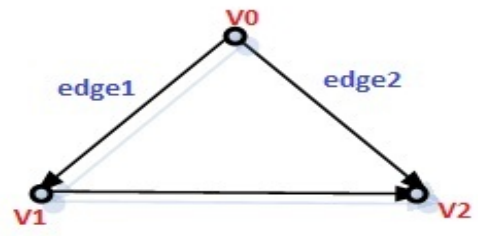

Figure 4.1: definition of edge1 and edge2.

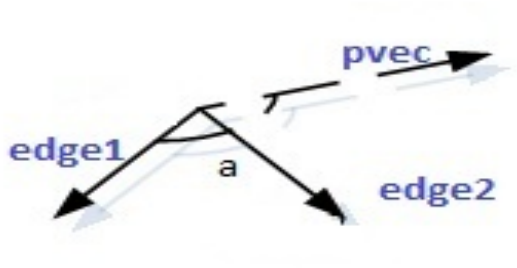

Figure 4.2: pvec.

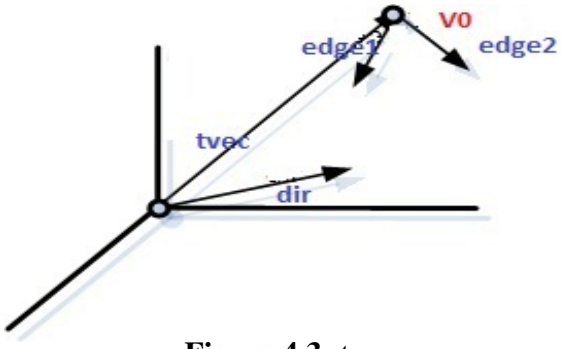

Figure 4.3: tvec.

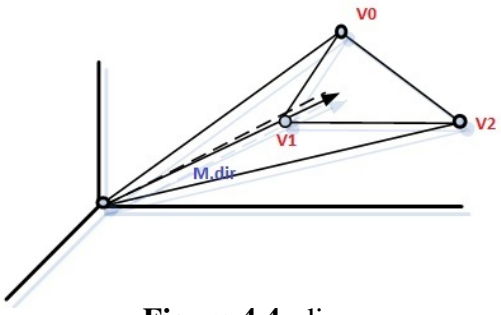

Figure 4.4: dir.

We first take the inner product of $\vec{pvec}$ and $\vec{dir}$ ($< pvec, dir >$). This gives the angle of intersection of $\vec{dir}$ to the plane Π containing the triangle. Another way to view this procedure is to note that this expression is equivalent to computing the determinant of $\vec{edge}1$, $\vec{edge}2$ and $\vec{dir}$. Thus, we are essentially testing for linear dependence of these three vectors. If $\vec{dir}$ is not linearly dependent to $\vec{edge}1$ and $\vec{edge}2$ (thus the observer is not on the plane Π), we have to see whether $\vec{dir}$ intersects the triangle or not. This is typically done using the barycentric coordinates, u,v and w. Informally, u, v and w

coordinates express the closeness of a point to vertices **v1**, **v2** and **v0** respectively. See figure 4.5 for the interpretation of u. Since $u+v+w=1$ we only need to compute u and v. If u,v and w are all in the range of 1 and 0 ($u,v,w \in [0,1]$), then the ray intersects the triangle, otherwise it does not. In the computation, we make use of the determinant of $\vec{dir}$, $\vec{tvec}$ and $\vec{edge}2$. Now, we show how the computations can be done for u, the case for v is similar.

Consider u=0. In this case, as we can see from Figure 4.6, the $\vec{edge}2$, $\vec{tvec}$ and $\vec{dir}$ are linearly dependent. So we can identify this case simply by checking the aforementioned determinant.

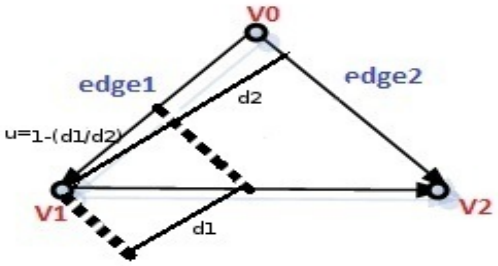

Figure 4.5: Geometric interpretation of the barycentric coordinate u.

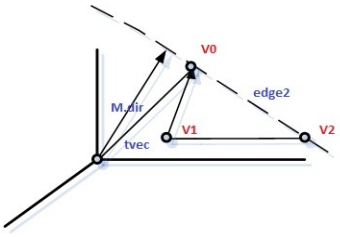

Figure 4.6: u=0.

Now consider u=1. In this case, the determinants of both $\vec{edge}2$, $\vec{tvec}$, $\vec{dir}$ and $\vec{edge}1$, $\vec{edge}2$, $\vec{dir}$ express the volume (up to a certain constant) shown in Figure 4.7, that is the pyramid with the triangle (v0,v1,v2) as the base and the peak at the viewpoint. So their ratio is 1. This means we can identify this case, using the ratio of the two aforementioned determinants.

$$D_1 = \begin{vmatrix} \vec{edge2} \\ \vec{tvec} \\ \vec{dir} \end{vmatrix} \tag{4.1}$$

$$D_2 = \begin{vmatrix} \vec{edge1} \\ \vec{edge2} \\ \vec{dir} \end{vmatrix} \tag{4.2}$$

$$u = \frac{D_1}{D_2} \tag{4.3}$$

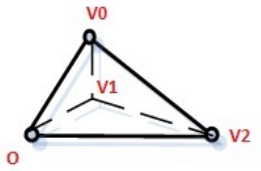

Figure 4.7: u=1.

Finally consider the case, 1>u>0. Here the ratio of the two determinants, related to the volume of the pyramids with bases $(v0'', v0', v2', v2''), (v0'', v0, v2, v2'')$ and the peak at the viewpoint are considered. It is easy to verify that this ratio equals u (see Figure 4.5) as well. One can also see this from the Figure 4.8.

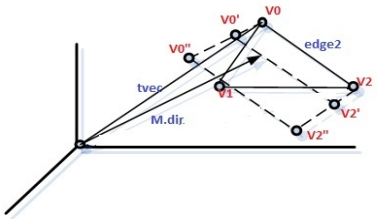

Figure 4.8: 1>u>0.

Using the above procedure to find intersection of the viewing ray and the faces in the scene, for each vertex in the scene, we create the viewing ray and compute the closest intersecting face for this ray. If this face is not adjacent to the vertex, we declare the

vertex invisible. If all the corners of a triangular face is invisible it's removed from the scene. Figure 4.9 illustrates the results of this process together with the original scene.

Figure 4.9: Face Culling. The processed scene, from the actual viewpoint(left) and a different(right) viewpoint.

It should be noted that this basic method is also an approximation, we can easily think of cases where this method would declare a face invisible, while some portion of it is actually visible. This problem shows itself more, when the variance of the face area distribution is high. This is, however, a practical solution to the problem; and the errenous visibility decisions rarely occur in scenes where faces are similar in size.

4.1.2 The approximate method

A naive implementation of the previously mentioned algorithm suffers from high computational complexity. Using Euler's formula for planar graphs, we can see that this operation has $O(v^2)$ complexity where v is the number of vertices. In practice, we see running times in the order of minutes for even medium size meshes. Practice confirms what the complexity stated above implies; the running times get much worse in bigger models. As a solution, we propose a novel method inspired by the simplification based surface to surface parameterization method called MAPS (Multiresolution Adaptive Parameterization of Surfaces) [23].

The MAPS method computes a surface to surface parameterization using a step by step simplification of the triangular mesh. At each step, a mapping from the original mesh to the simplified mesh is computed. With this mapping (inverse of parameterization function), we can cull faces of the simplified mesh and declare vertices invisible if either

1. the corresponding vertex in the simplified mesh is declared invisible or,

2. the vertex is mapped to an invisible face in the simplified mesh.

Note that the choice of the simplification procedure is not arbitrary but essential in order not to remove visible vertices from the scene, thus creating artifacts. The simplification method which we have used is half-edge collapse with Garland's quadric error metric [24]. We have chosen this metric in order to keep vertices in non-smooth regions intact (i.e. during simplification, we try not to remove faces that will hide other faces with a high probability), so that visible regions for the original and simplified meshes are similar. This is why, in practice, we have never observed the error of removing visible vertices from the scene. We conjecture that such an error is possible only, i) at a very high simplification level or ii) in pathological cases (i.e. models that are not used in practice). The error of keeping a vertex when, in fact, it is invisible is the typical error that occurs in the explained procedure. The parameterization of a single vertex consists of three steps:

1. The vertex to be removed during simplification and all neighboring vertices are mapped to a planar polygon via the discrete conformal mapping in MAPS (Fig. 4.10).

2. The vertex is removed and the resulting hole is triangulated (the possible triangulation is unique in our case since we only consider the half-edge collapse operation).

3. The triangle which contains the removed vertex in the new triangulation is determined and the barycentric coordinates are computed(Fig. 4.11). This coordinate is used to find the mapping of the removed vertex to the simplified surface.

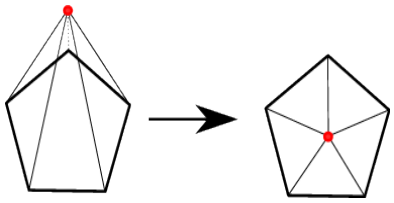

Figure 4.10: Mapping from the surface to the planar polygon.

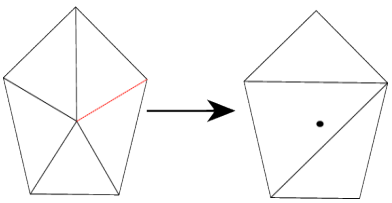

Figure 4.11: Half-edge collapse and the computation of the barycentric coordinates for the removed vertex.

We can easily see that such a procedure will result in some of the invisible vertices deemed visible after culling but Figure 4.12, that presents run time vs number of removed vertices, shows that the speed advantage is considerable (the mesh originally has 2832 vertices). The rightmost point on the graph corresponds to the original method (without simplification).

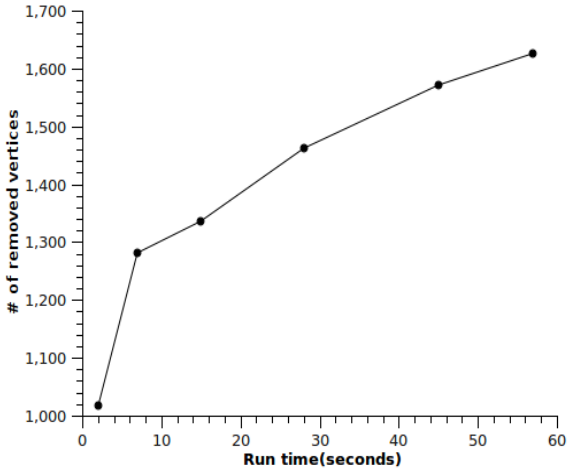

Figure 4.12: Run time vs number of the vertices removed.

The main advantage of such a scheme becomes obvious when the same parameterization that is used in the culling process is also employed in the coding process (such as wavelet transform based coding systems [25]). In this case, the additional cost of the culling operation becomes almost zero.

The choice of the operating point is independent of the rate allocation method. Thus, the discussion in the subsequent chapter does not assume a particular choice of simplification level used for face culling.

4.2 The Effect of Quantization Noise

So far, the problem of removing invisible vertices from a scene according to a particular viewpoint. Thus, the discussion so far is generic, in the sense that the particular application of this procedure is not considered. However, in our case, the application must be considered, since the processed scene will not be used as is, but quantized and transmitted to the end user before being used. The error due to the transmission can be ignored with the help of advanced error correcting/detecting codes and transmission protocols like TCP. However, the quantization noise is still an important factor to be considered. Previous work on view-dependent coding omit this, but it turns out to have a huge impact on the resulting scene quality, since the quantization noise perturbs the vertex positions and thus may open up "cracks" in the scene. Figure 4.13 shows an example.

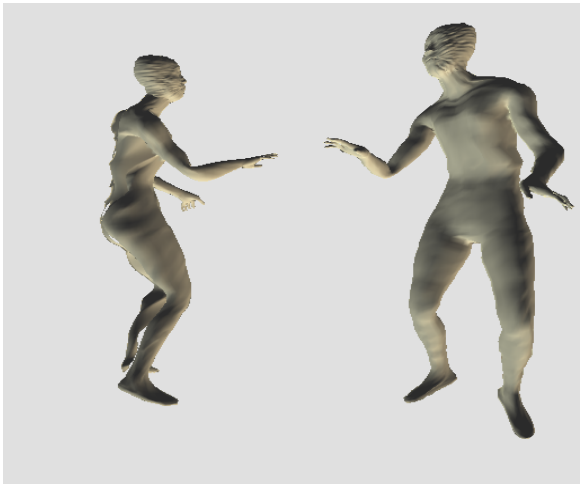

Figure 4.13: An example of artifacts due to quantization of culled scenes.

The artifacts can be "fixed" using different approaches. The method employed in this Thesis is to add an additional layer of vertices, which are the invisible vertices neighbouring visible ones. This approach enjoys simplicity of implementation and small computational burden. However, much of the vertices added this way can be invisible in the culled scene as well. For example, in the "Dancers" scene, this additional layer consists of almost 10% the number of vertices in the culled scene, thus the addition of these lowers the coding performance. Another solution would be to code these additional vertices in a selective fashion using local prediction schemes.

Thus bits would be allocated only to vertices which will be visible in the resulting scene. This approach will probably be more efficient than the current method, but would possibly suffer from a high computational burden. More detailed comparison of these two approaches is beyond the scope of this Thesis and will be done in another publication.

5. VIEW DEPENDENT RATE-ALLOCATION

Based on the discussion of rate allocation methods in Chapter 3, we can solve the rate-distortion optimization problem given a bit budget and distortion measure. Such a measure should capture the visual quality of the scene while being easy to compute. In this chapter, several such distortion measures (of varying quality and complexity), that were previously reported in the literature as well as the novelly proposed in this Thesis, are discussed. We assume a fixed viewpoint scenario to illustrate the methods.

First, we introduce the several previously reported distortion measures for rate allocation using the basic framework described and after a comparison of the methods using a sample scene, we propose a hybrid method that takes advantage of various properties of the basic methods.

5.1 View-dependent Rate Allocation Methods

The methods discussed in this chapter can be discriminated based on whether they allow fast re-computation ($O(|J|)$), where $|J|$ is the number of regions and J is the index set of regions in the scene, of the allocation results when the viewpoint changes.

The results are discussed without reference to the coding method employed, since we assume that the view-dependent explicit rate allocation methods are independent of the particular choice of mesh compression method. In this Thesis, we have used the method of spectral transform [13] together with CSPECK [14, 12] for transform coefficient coding. The use of CSPECK method in color image coding provides embeddedness and progresiveness. The embeddedness property enables the collection of all distortion measures at various rates in a single coding pass, instead of coding separately for each rate. However, since we are using an explicit rate allocation among different regions (quantizers) for view dependent coding and the resulting bit-streams for each region are concatenated sequentially, the resulting bitstream is not truly embedded.

In [4, 5] the view dependendent rate allocation methods are considered for allocating rate to regions of the geometry of a single 3-D mesh. In this work, the more general problem of rate allocation to multiple 3D meshes is considered. Therefore, the methods of [4, 5] were extended by incorporating distance into their measures in accordance with [6].

5.1.1 Image based method

In this method, a MxN pixel image of the reconstructed scene is rendered and 2D MSE is computed as the image distortion measure.

$$D_{image} = \frac{1}{M \cdot N} \sum_{i \in [0,N-1], j \in [0,M-1]} (I_l(i,j) - I'_l(i,j))^2 \qquad \textbf{(5.1)}$$

Here I is the luminance function for the original image, and I' is the luminance function for the rendered image of the reconstructed scene. $I_l(i,j)$ is the luminance value at the (i,j) coordinates of the image.

The distortion in the rendered image due to compression has the same effect on all color channels. Hence, we have chosen to evaulate distortion using only luminance and discarded the chrominance values.

When distortions at all operating points are estimated with this method, the resulting rate allocation scheme is not feasible due to the high computational complexity. Therefore, the results reported in 5.1.5 for this method is used for the purpose of benchmarking. In Section 5.1.6, we present a combination of the image rendering based method with the distance based method, which has a higher efficiency than the former, with close performance.

Note that in this work, we have used “image based method” and “image rendering based method” interchangeably.

5.1.2 Distance based method

This method was proposed in [6]. The distortion measure for each region is the MSE of the vertex coordinates weighted by the inverse squared average distance of vertices in the mesh to the viewpoint.

$$D_{dist,j} = \frac{1}{d_j^2} \cdot D_{mse,j} \tag{5.2}$$

$$D_{dist} = \sum_{j \in J} D_{dist,j} \tag{5.3}$$

In the above definition, J is the set of indices of all regions in the scene, d_j is the average distance of the points in jth region to camera. $D_{mse,j}$ is the MSE (in 3D space) distortion of the vertices in the region with index j.

Although a conceptually simple approach, the distance based rate allocation offers a robust solution under different rendering conditions and coding rates. It is also the fastest to compute and recompute. Asymptotically, the recomputation complexity is $O(|J|)$.

5.1.3 Visibility based methods

The visibility based methods were proposed in [4]. The idea is to allocate more bits to more visible regions. Visibility of a vertex is defined in [4] as the cosine of the angle between viewing direction ($\vec{V}$) and vertex normal ($\vec{n}$). This visibility value, together with the distance of the vertex, is used to weigh the squared error ($||\vec{v} - \vec{v}'||)^2$) of the vertex.

$$D_{visslow} = \sum_{i \in I} ((\vec{n}_i \cdot \vec{V})^2 \cdot \frac{(||\vec{v}_i - \vec{v}_i'||)^2)}{d_i^2} \tag{5.4}$$

A lower complexity variant of 'visibility' is also defined for a region as the cosine of the angle between the average normal of the region ($\vec{\vec{N}}$) and the viewing direction.

$$D_{visfast} = \sum_{j \in J} (\vec{\vec{N}}_j \cdot \vec{V})^2 \cdot D_{dist,j} \tag{5.5}$$

In the above definitions, I is the set of all vertices on the scene and d_i is the distance of ith vertex to viewpoint.

The fast variant, unlike the slow variant, allows recomputation of the the error values for each region in $O(|J|)$ where $|J|$ is the number of regions in the mesh and is much smaller than $|I|$.

In our experiments, we have noticed that the performance of the fast visibility based method falls short of any other method, so we have tried another approach based on the same idea. Instead of estimating a value that is proportional to the visibility of regions, the new method, called exact visibility based method, computes the exact visible area for each region. The computation is done by assigning each region a unique color and rendering the scene with these colors only (without lighting) and counting the number of pixels in the scene for each region.

$$D_{visexact} = \sum_{j \in J} a_j \cdot D_{mse,j} \tag{5.6}$$

In the above definition, a_j is the area, in pixels, of the region with index j.

5.1.4 Screen space distortion based methods

Methods taking screen space distortion (SSD) and illumination conditions into account were proposed in [5]. The idea is that the visible distortion on the scene is mainly due to visible coding error (i.e. those orthogonal to the viewing vector) and errors in the normals which affect the shading results. The methods proposed in [5] contain terms for both specular and diffuse lighting components. Here, we only consider the diffuse term (distortion of normals) in order to be able to compare this with other methods. We consider a method that integrates Screen Space Distortion and Normal Distortion as well as a simpler variant that is based only on Screen Space Distortion.

$$D_{screen} = \sum_{i \in I} \frac{1}{d_i^2} \left|\left| (\vec{V} \times (\vec{v}_i - \vec{v}_i')) \right|\right|^2 \tag{5.7}$$

$$D_{norm} = \sum_{i \in I} \frac{1}{d_i^2} \left|\left| (\vec{n}_i' - \vec{n}_i) \right|\right|^2 \tag{5.8}$$

$$D_{screenandnorm} = \lambda \cdot D_{screen} + D_{norm} \tag{5.9}$$

In the above definitions, n_i and n_i' are the actual and reconstructed normals for ith vertex, respectively.

Note that the two factors are combined using an arbitrary coefficient λ. A poor choice of this value leads to poor coding performance.

One can easily see that the complexity of the methods are both $O(|I|)$. The addition of the normal distortion term usually yields a significant improvement over the screen space distortion based method if the λ is properly selected. However, the use of the $\lambda \cdot D_{screen}$ term requires an additional optimization procedure, since the optimal value varies from scene to scene.

5.1.5 Comparison

In this section, we shall make a brief discussion of the various methods described earlier, using a sample scene. The conclusions drawn on this scene generalized well to the other scenes that we experimented with. In Figure 5.1 we present the original "Dancers" scene.

In Table 5.2, we report the performances of the methods for this scene using Peak Signal-to-Noise Ratio (PSNR) measure in dB (the methods that have lower complexity than $O(|J|)$ are placed under category A). For all of the rate-allocation methods, we collect distortion measurements using steps of 0.2 bpv from 3 bpv to 11 bpv. This range was determined experimentally during the tests, considering both the final scene quality and the run time of the methods.

Table 5.1: PSNR values at rate 4.5 bpv for various view-dependent rate-allocation methods.

Category	Method	PSNR
A	Distance Based	31.19
	Visibility Based (fast)	29.18
	Visibility Based (exact)	30.83
B	Image Based	31.94
	SSD and Normal Distortion	31.44
	Screen Space Distortion(SSD) Based	31.53
	Visibility Based (Slow)	31.03

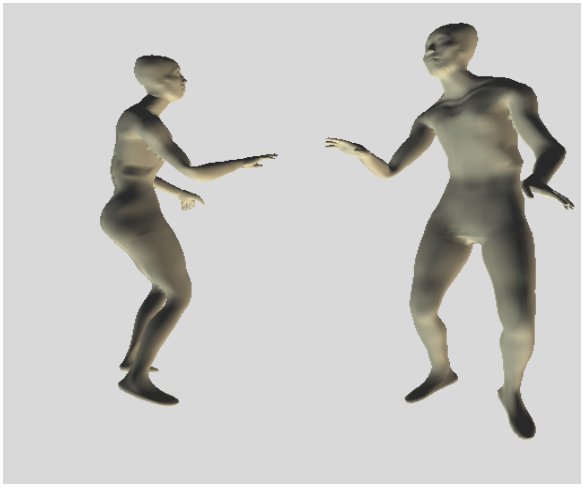

Figure 5.1: The Dancers Scene.

The performances of the various view-dependent rate allocation methods are compared at various rates (using the same scene) in Figure 5.2 .

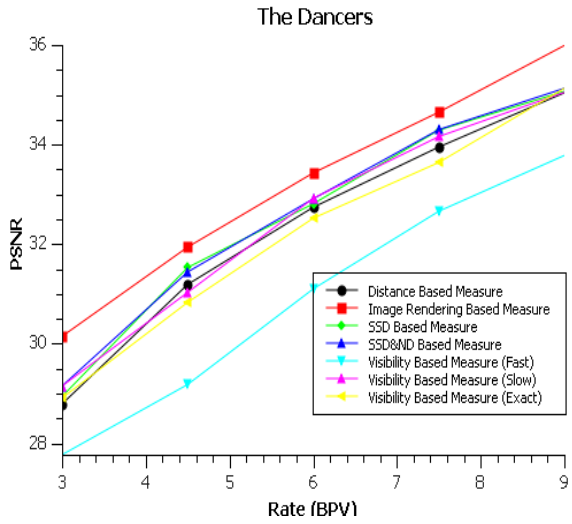

Figure 5.2: PSNR vs Rate for various view-dependent rate-allocation methods.

Among the A category, we notice that distance based method gives a very consistent and good performance, while the fast variant of the visibility method falls short almost everywhere. In the B category, the image based method wins in terms of PSNR as expected. However, considering the large difference in terms of complexity, the success of the SSD with normal distortions method is surprisingly close to it. Note that this performance is achieved only when the λ is properly selected. Since the optimal value is different for different scenes, we end up with another optimization

problem which makes this method even more time-consuming than the image based one in the end.

Using only the screen space distortion also yields very good performance and this result also applies well to other scenes and rendering conditions. The most surprising result of the experiments is that the distance based method, being by far the fastest one, also is a very good competitor even with category B methods in terms of performance.

There is an immediate conclusion to be drawn from these results; the distance based method comes very close to other methods at any rate with small complexity and posesses the capability of updating the allocation when the viewpoint is moved. This observation leads to the idea that if we can come very close to the results of the image based method with a much faster method like the distance based one, we might use the fast method for an initial allocation and improve the results with the image based method. This approach is discussed next.

5.1.6 The hybrid method

In Chapter 3, the problem of rate allocation was defined as an optimization problem. The discussion progressed in terms of the Lagrange multipliers based solution to the problem. Here, an alternative view of the optimization problem is presented.

Recall the constrained optimization problem (**3.1**). The constraint term $\sum_{i=0}^{n} b_i \leq R_c$ corresponds to the limited bit budget. The inequality is present to handle the case where, increasing the rate in any of the quantizers results in an increase in distortion (contrary to a decrease). However, this case rarely (if ever) happens in practice. Even if it is possible for a single region to exhibit such a behaviour at a given rate, we have not encountered a single case where all regions behaved like this, at a given rate vector. Thus, we take the constraint as $\sum_{i=0}^{n} b_i = R_c$.

In this case, the set of admissible rate vectors constitutes a bounded hyperplane $\mathscr{P}$ such that the normal is in the direction of $(1,1,1,...)$ and $\mathscr{P}$ lies in the first quadrant. Then we can view the method described in Chapter 3 as a walk that starts from the origin and proceeds by moving in the direction of the axis that has the maximum $\frac{\Delta D}{\Delta R}$ (so that each step is either parallel or orthogonal to the previous one). The point at which the walk

intersects $\mathscr{P}$ is taken as the optimal rate vector. The process is illustrated in Figure 5.3. The steps are color coded in the figure to indicate the relative gain ($\frac{\Delta D}{\Delta R}$), where red means a large gain and green means a small one. The axes indicate the rate assigned to a particular quantizer.

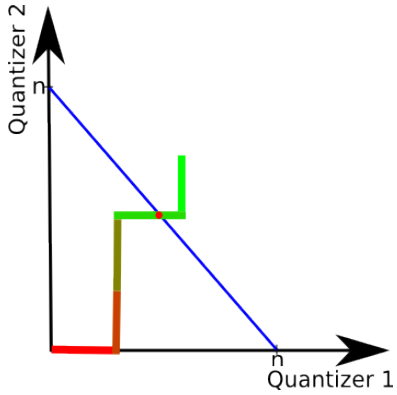

Figure 5.3: Illustration of the rate allocation process, with two regions (quantizers) in the scene. The blue line indicates the set of admissible points. The red dot indicates the optimal rate vector.

Now with this view in mind, we can think of other methods for the optimization of $D(\vec{b})$ (total distortion of the scene, when coding is performed with $\vec{b}$ as the rate vector). Taking an initial point in $\mathscr{P}$, one can optimize using any popular optimization technique. However, in our case we can very quickly compute a good initial point that is known to be close to the optimal, that is we can use the distance based method's result as an initial point. We, therefore, suggest a simple greedy approach. The proposed method, called the hybrid method, works by sampling $D(.)$ in a fixed neighbourhood of the current rate vector $\vec{b}$, lying on $\mathscr{P}$. This neighbourhood forms a grid-like structure in the solution space. The point which reduces the distortion most is taken as the current rate point and the process is reiterated. The method stops when no neighbourhood provides a better alternative.

The "Animal Kingdom" and the "ThreeKings" scenes (Figure 5.4 and 5.6) is used to compare the distance based, image based and the hybrid method's performances. Figure 5.5 and 5.7 shows the resulting PSNR values for different rates.

Figure 5.4: The Animal Kingdom Scene.

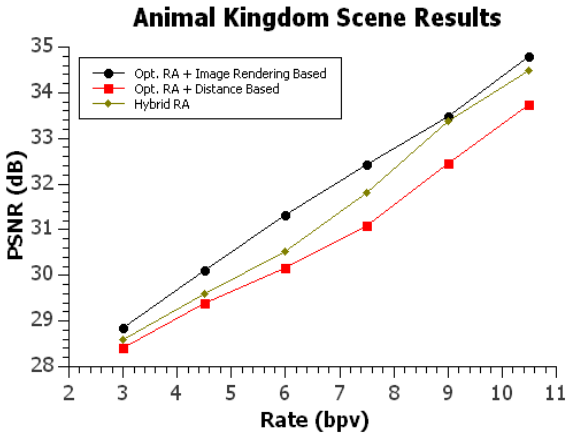

Figure 5.5: PSNR vs. Rate for the Animal Kingdom Scene.

Figure 5.6: The Three Kings Scene.

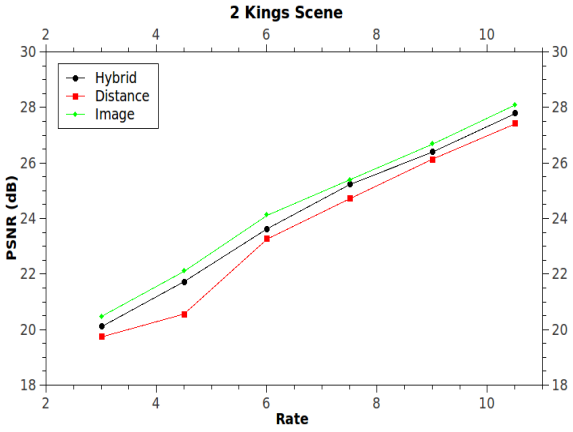

Figure 5.7: PSNR vs. Rate for the Three Kings Scene.

Table 5.2: Running time of the rate allocation methods for various scenes at a rate of 9.

Scene	Image Based	Distance Based	Hybrid
Animal Kingdom	246.75	3.24	6.8
MyFans	412.71	10.09	10.54
TheHorse	353.22	7.78	9.5
TheDancers	220.39	4.83	6.04

In order to better evaluate the speed advantage of the hybrid method over pure image based one, we present the following table, showing execution time (in seconds) of the three methods at a rate of 9 for different scenes.

Finally, two more scenes are shown and used to compare the methods discussed in the literature with the two newly proposed method of this Thesis. First, the "MyFans" scene is shown in Figure 5.8. The results are shown in Figure 5.9. The difference in performance of different methods is more clear in this scene. Among the methods previously proposed in the literature, the Screen Space Distortion Based methods comes closest to the newly proposed methods at the rate of 6 bits/vertex. Figure 5.10 shows a qualitative comparison of the resulting images between the hybrid method and the SSD&N distortion based method. In this figure we can see that the hybrid method is capable of preserving the planar regions intact, with small distortion. The method is able to make use of the contrast differences among regions, such that a very bright region where the distortion is not as visible is used to tradeoff the quality with

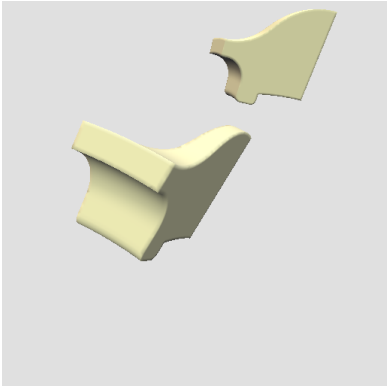

Figure 5.8: MyFans scene, showing the fandisk mesh in different orientation and positions.

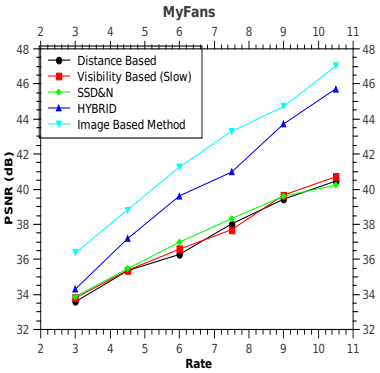

Figure 5.9: MyFans scene results.

regions which have a bigger impact on the visible distortion. Of course, in another scene, the same tradeoff would also occur in darker regions. The seamless integration of visibility of regions, visibility of the quantization noise and other factors such as lighting conditions enable the image based and the hybrid methods to be superior to other methods in the literature. The PSNR results can be found in Figure 5.9.

The final figure is different than the rest in the aspect that it is a scene composed of a single highly detailed mesh instead of multiple models. Figure 5.11 shows this scene. This scene is provided in order to demonstrate the efficiency of the proposed methods even for a single mesh. Figure 5.12 shows the results.

Appendix A contains further discussion on the conditions of optimality for the proposed method.

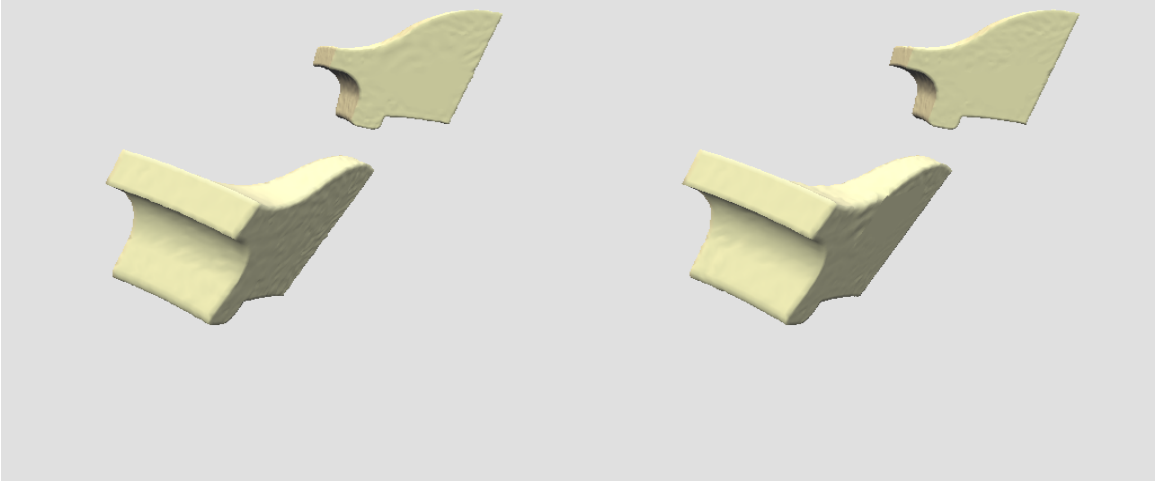

Figure 5.10: A visual comparison of the two methods: on the left SSD&ND method and on the right the Hybrid method.

Figure 5.11: The Horse scene, containing a single large model.

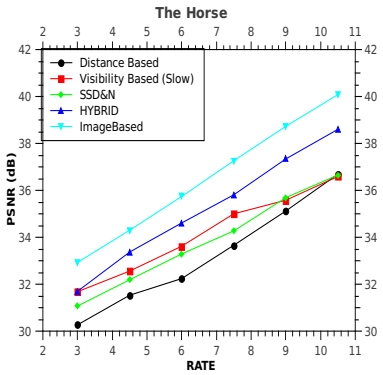

Figure 5.12: The Horse scene results.

6. CONCLUSION AND FUTURE WORK

In this Thesis, we considered several approaches to solve the rate-allocation problem in a view dependent manner. Having discussed various advantages and disadvantages of the previously proposed methods, and compared them to the practical upper bound on performance, we have shown that it is possible to devise a method that is both fast and better in performance than others. However, it is also shown that the proposed practical method still falls short of the optimal allocation. The performance gap between the optimal rate allocation with image rendering based measure and the proposed fast method is largely due to the nonconvexity of the operational rate-distortion curves and the local search characteristic of the improvement stage of the hybrid method.

From this point on, several improvements are possible, first one can try global optimization schemes instead of the local greedy method used in this work. Another extension would be to use a better distortion metric that would better relate to the human visual system, than MSE.

REFERENCES

[1] **Bo, L. and Hongbin, Z.**, (2004). Scalable mesh coding based on wavelet transform, *Signal Processing, 2004. Proceedings. ICSP '04. 2004 7th International Conference on*, **2**, 1139 – 1142 Vol.2.

[2] **Stefanoski, N. and Ostermann, J.**, (2010). SPC: Fast and Efficient Scalable Predictive Coding of Animated Meshes, *Computer Graphics Forum*, **29(1)**, 101–116.

[3] **Yang, S.**, **Shen, M. and Kuo, C.**, (2004). Progressive coding of 3D textured graphic model via joint mesh-texture optimization, *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, **5**, V – 945–8 Vol.5.

[4] **Sim, J.**, **Kim, C.**, **Kuo, C. and Lee, S.**, (2005). Rate-Distortion Optimized Compression and View-Dependent Transmission of 3D Normal Meshes, *IEEE Transactions on Circuits and Systems for Video Technology*, **(7)**, 854–868.

[5] **Guan, W.**, **Zhang, J.**, **Cai, J. and Zheng, J.**, (2007). Illumination and View Dependent Progressive Transmission of Semi-Regular Meshes with Subdivision Connectivity, *Computer graphics and interactive techniques in Australia and South East Asia*, 219–226.

[6] **Payan, F.**, **Antonini, M. and Meriaux, F.**, (2009). View-Dependent Geometry Coding of 3D Scenes, *Proceedings of IEEE International Conference in Image Processing (ICIP)*.

[7] **Reddy, M.**, (2001). Perceptually Optimized 3D Graphics, *IEEE Computer Graphics and Applications*.

[8] **Luebke, D. and Hallen, B.**, (2001). Perceptually Driven Interactive Rendering, *Tech Report, University of Virginia*.

[9] **Sayood, K.**, (2000). Introduction to data compression, Morgan Kaufmann series in multimedia information and systems, Morgan Kaufmann Publishers.

[10] **Berger, T.**, (1971). Rate distortion theory: a mathematical basis for data compression, Prentice-Hall series in information and system sciences, Prentice-Hall.

[11] **Ben-Chen, M. and Gotsman, C.**, (2005). On the optimality of spectral compression of mesh data, *ACM Trans. Graph.*, **24**, 60–80.

[12] **Bayazıt, U.**, **Konur, U. and Ateş, H.**, (2010). 3-D Mesh Geometry Compression with Set Partitioning in the Spectral Domain, *IEEE Transactions on Circuits and Systems for Video Technology*, **20(2)**, 179 – 188.

[13] **Karni, Z. and Gotsman, C.**, (2000). Spectral Compression of Mesh Geometry, *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 279 – 286.

[14] **Pearlman, W.**, **Islam, A.**, **Nagaraj, N. and Said, A.**, (2004). Efficient, Low-Complexity Image Coding with a Set-Partitioning Embedded Block Coder, *IEEE Trans. Circuits and Systems for Video Technology*, **14**, 1219–1235.

[15] **Shoham, Y. and Gersho, A.**, (1988). Efficient Bit Allocation for an Arbitrary Set of Quantizers, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **36(9)**, 1445–1453.

[16] **Alliez, P. and Desbrun, M.**, (2001), Valence-Driven Connectivity Encoding for 3D Meshes.

[17] **Gotsman, C.**, (2003), On the Optimality of Valence-based Connectivity Coding.

[18] **C., T. and C., G.**, (1998). Progressive coding of 3D textured graphic model via joint mesh-texture optimization, *Proceedings of Graphics Interface, Vancouver*.

[19] **Payan, F. and Antonini, M.**, (2004). Model-based Bit Allocation for Normal Mesh Compression, *Proceedings of IEEE international workshop on MultiMedia Signal Processing (MMSP)*.

[20] **Ortega, A. and Ramchandran, K.**, (1998). Rate-Distortion Methods for Image and Video Compression, *IEEE Signal Processing Magazine*, 23 – 50.

[21] **Riskin, E.**, (1991). Optimal Bit Allocation via the G-BFOS Algorithm, *IEEE Transactions on Information Theory*, **37(2)**.

[22] **Möller, T. and Trumbore, B.**, (1997). Fast, Minimum Storage Ray-Triangle Intersection, *Journal of Graphics Tools*, **2**, 21–28.

[23] **Lee, A.**, **Sweldens, W.**, **Schröder, P.**, **Cowsar, L. and Dobkin, D.**, (1998). MAPS: Multiresolution Adaptive Parameterization of Surfaces, *Proceedings of SIGGRAPH 98*.

[24] **Lee, A.**, **Garland, M. and Heckbert, P.**, (1997). Surface Simplification Using Quadric Error Metrics, *Proceedings of SIGGRAPH 97*.

[25] **Khodakovsky, A. and Guskov, I.**, (2002). Normal Mesh Compression, *Geometric Modeling for Scientific Visualization, Springer-Verlag*.

[26] **Alper, M.E. and Bayazıt, U.** A Fast Image Based View Dependent Rate-Allocation Method for 3-D Scenes, *will be submitted to Transactions on Multimedia*.

APPENDICES

APPENDIX A.1 : OPTIMALITY OF HYBRID METHOD

Appendix A.1: Optimality of the proposed method

The derivations in this section are due to Uluğ Bayazıt. These results are also reported in [26].

Let $J_i^* = D_i^* + \lambda R_i^*$ and $J_i^m = D_i^m + \lambda R_i^m$ be the costs of the optimal rate allocation and rate allocation after m'th step of the improvement stage for i'th source(region) and define $\Delta J_i^m = J_i^* - J_i^m$, $\Delta R_i^m = R_i^* - R_i^m$. We partition regions into groups with indices $I^{+,m}$ and $I^{-,m}$ depending on the sign of ΔR_i^m, i.e. $I^{+,m} = \{i : \Delta R_i^m > 0\}$ and $I^{-,m} = \{i : \Delta R_i^m < 0\}$. Define further $\Delta J^{g,m} = \sum_{i:i\in I^{g,m}} J_i^m$, $\Delta R^{g,m} = \sum_{i:i\in I^{g,m}} R_i^m$ for $g \in \{+,-\}$. It must be that $\Delta J^{+,m} < -\Delta J^{-,m}$ since at m'th step we have not yet arrived at the optimum allocation. Since $\Delta R^{+,m} + \Delta R^{-,m} = 0$

$$\frac{\Delta J^{+,m}}{\Delta R^{+,m}} < -\frac{\Delta J^{-,m}}{\Delta R^{+,m}} = \frac{\Delta J^{-,m}}{\Delta R^{-,m}}$$

From two groups, we select one source each with indices

$$i^{*,+} = \arg\min_{i:i\in I^+} \frac{\Delta J_i^m}{\Delta R_i^m}, i^{*,-} = \arg\min_{i:i\in I^-} \frac{\Delta J_i^m}{\Delta R_i^m}$$

Due to their definitions, the sources with these indices satisfy

$$\frac{\Delta J_{i^{*,+}}^m}{\Delta R_{i^{*,+}}^m} < \frac{\Delta J_{i^{*,-}}^m}{\Delta R_{i^{*,-}}^m}$$

Now, if we assume that the operational rate-distortion characteristic for each source is convex, i.e. $\frac{\delta D_i^m}{\delta R_i^m} > 0$ then $\frac{\delta J_i^m}{\delta R_i^m} > 0$. This in turn implies that $\frac{\Delta J_{i^{*,-}}^m}{\Delta R_{i^{*,-}}^m} < \frac{\delta J_{i^{*,-}}^m}{\delta R_{i^{*,-}}^m}$ and $\frac{\delta J_{i^{*,+}}^m}{\delta R_{i^{*,+}}^m} < \frac{\Delta J_{i^{*,+}}^m}{\Delta R_{i^{*,+}}^m}$. Hence we get

$$\frac{\delta J_{i^{*,+}}^m}{\delta R_{i^{*,+}}^m} < \frac{\delta J_{i^{*,-}}^m}{\delta R_{i^{*,-}}^m}$$

The above equation suggests that by selecting $\delta R_{i^{*,-}}^m = -\delta R_{i^{*,+}}^m = \varepsilon$ we can achieve a change (reduction) in total cost of $\delta J_{i^{*,+}}^m - \delta J_{i^{*,-}}^m$ at the m'th step, if we have not yet reached the optimum allocation with the least total cost.

CURRICULUM VITAE

Candidate's full name:	Muzaffer Ege ALPER
Place and date of birth:	Istanbul, 02 October 1985
Universities and Colleges attended	Istanbul Technical University
Address:	ITU Ayazaga Campus, Faculty of Computer and Informatics, Sarıyer, İstanbul
Phone:	0212 285 38 52
email:	malper@itu.edu.tr

PUBLICATIONS:

- Image Based Rate-Allocation for 3D Scenes, *SIU 2010 Diyarbakır*
- Improving Course Success Prediction using ABET Course Outcomes and Grades *CSEDU 2012*