ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

VIEW-DEPENDENT CODING OF ANIMATED MESH SEQUENCES

M.Sc. THESIS

Semih ÇELİK

Department of Computer Engineering

Computer Engineering Programme

AUGUST 2014

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

VIEW-DEPENDENT CODING OF ANIMATED MESH SEQUENCES

M.Sc. THESIS

Semih ÇELİK

Department of Computer Engineering

Computer Engineering Programme

Thesis Advisor: Prof. Dr. ULUĞ BAYAZIT

AUGUST 2014

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

CANLANDIRILMIŞ GÖZ DİZİLERİNİN BAKIŞ NOKTASI BAĞIMLI KODLANMASI

YÜKSEK LİSANS TEZİ

Semih ÇELİK
(504111529)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Uluğ BAYAZIT

AĞUSTOS 2014

Semih ÇELİK, a M.Sc. student of ITU Graduate School of Science Engineering and Technology **504111529** successfully defended the thesis entitled **"VIEW-DEPENDENT CODING OF ANIMATED MESH SEQUENCES"**, which he prepared after fullfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Uluğ BAYAZIT** Istanbul Technical University	.....
Co-Advisor :		.....
Jury Members :	**Assoc. Prof. Hazım Kemal EKENEL** Istanbul Technical University	.....
	Assoc. Prof. Hakan GÜRKAN Işık University	.....
		.....
		.....
		.....

Date of Submission : 1 August 2014
Date of Defense : 5 August 2014

To my family,

FOREWORD

Thanks to my thesis advisor Uluğ BAYAZIT, who represent the subject to me and took my attention.
Thanks to TUBITAK, who support my education with scholarship.

August 2014 Semih ÇELİK

TABLE OF CONTENTS

Page

ABBREVIATIONS

bpvf	**:** Bit Per Vertex Frame
PCA	**:** Principal Component Analysis
R-D	**:** Rate - Distortion
VDC	**:** View Dependent Compression
VIC	**:** Viewpoint Independent Compression

LIST OF TABLES

Page

LIST OF FIGURES

Page

CANLANDIRILMIŞ GÖZ DİZİLERİNİN BAKIŞ NOKTASI BAĞIMLI KODLANMASI

ÖZET

Canlandırılmış (dinamik) göz dizi modelleri 3 boyutlu model yüzeylerinin görselleştirilmesi ile üç boyutlu cisimlerin temsilinde sıklıkla kullanılmaktadır. Canlandırılmış sentetik nesnenin hareketi ve diğer değişimler nesne yüzeyinin değişimi ve hareketi ile ifade edilmektedir. Hareketli göz dizi modellerini ifade etmek için gerekli bit sayısı statik göz modellerine kıyasla oldukça fazladır. Video ve resim sıkıştırma arasındaki fark gibi static göz modellerini sıkıştırma ve dinamik göz dizi modellerini sıkıştırma yöntemleri arasında farklılık doğmaktadır. Bu nedenle dinamik göz dizi modellerinin sıkıştırılması üç boyutlu grafik ile ilgili alanlar için önem taşımaktadır.

Bakış noktası tarafından görülen bölgenin belirlenmesi, göz dizi modelleri üzerinde yapıan birçok işlemde kullanılmaktadır. Göz dizi modellerini bilgisayar ortamında görselleştirme ve fizik testleri (nesne çarpışmaları, birbirlerine etkileri gibi) gibi işlemlerde kullanılan bakış noktası tarafından görünürlülük testi işlemi sıkıştırma yöntemleri tarafından da kullanılmaya başlanmıştır.

Bakış noktası temelli sıkıştırma, yani belli bir bakış noktasından görünmeyen kısımları kodlamayarak sadece bakış noktasından görünen bölgenin kodlanması, static göz modelleri için yakın zamanlı bazı çalışmalarda ele alınmıştır. Görünmeyen bölgenin kodlanmamasına dayanmayan fakat, yine bakış noktası bağımlı olan ve bakış noktasından görünmeyen kısımları daha az bitle dolayısıyla daha fazla kayıpla kodlayan sıkıştırma yöntemleri incelenmiştir. Bu çalışmalar bakış noktasına görünmeyen bölgeleri tamamen çıkartmamakla birlikte bu bölgeler için harcanan bitleri azaltarak görünür hataları arttırmadan (ama görünmeyen bölgede hataları artmasına sebep olarak) dinamik göz dizi modellerini daha az bitle sıkıştırmaktadır.

Bu çalışmada, benzer çalışmalarda da varsayılan, alıcı tarafındaki kullanıcı bakış noktası bilgisinin verici tarafında bilindiği varsayımı kabul edilerek, canlandırılmış göz dizi modellerinin sadece görünen yüzey bölgesindeki düğümlerin kodlanması önerilmiştir. Kodlanan bölgenin sınırlandırılması ile kodlanan düğüm sayısınn basit bir varsayımla yarı yarıya azalacağı varsayılmıştır. Bu bağlamda, bit gereksinimini düşürecek temel varsayım zaman içinde (çerçeveden çerçeveye) bakış noktası tarafından görünen yüzey bölgelerinin büyük değişimler göstermememsidir. Bu varsayım sonucunda ardışık çerçevelerin ikisinde de görünen bölgede kalan düğüm sayısının, ardışık bölgelerde görünür bölgeye giren yada çıkan düğüm sayısından çok büyük olması beklenmektedir. Önerilen sistemde bakış noktasından görünmeyen kısımların atılması sonucunda, önceki çerçevede görünmeyen ancak sıkıştırılmak üzere olan çerçevede görünen bölgeye giren düğümler olacaktır. Bu düğümlerin zamansal öngörü ile kodlanması mümkün olmadığından uzamsal öngörü ile kodlanacaktır. Ayrıca kodlanan göz dizilerinde değişen görünürlülük bilgisinin alıcı

tarafına gönderilmesi gerekmektedir. Farklı bir söyleyişle, önerilen sistemde kodlama için gerekli bit miktarını değiştiren üç etken vardır.

1. Sıkıştırılacak olan çerçevede görünmeyen düğümleri kodlamayarak gereken bit miktarı düşürülmektedir.
2. Ardışık çerçevelerde görünürlükteki değişimi belirtmek için bir miktar bit gereklidir.
3. Bir önceki çerçevede görünmeyen, ancak sıkıştırılmakta olan çerçevede görünür olan düğümler bulunmaktadır. Bu düğümler için zamansal olarak ilişkilendirme yapılarak, konumu için daha iyi öngörü yapılmasını sağlayan önceki çerçeveye ait düğümler önceki çerçevede kodlanmamıştır. Bu nedenle, zamansal (çerçeveler arası) ilişkilendirme yapılamaz. Sadece uzamsal ilişkilendirme (çerçeve içi) yapılması, düğüm noktasının mevcut konumu için öngörü hesaplayan algoritmasının performansının düşüşüne sebep olacaktır. Bu performans kaybının sebep olduğu bir miktar bit artışı vardır.

İlk iki etken direk olarak bit miktarını etkilemekte ancak üçüncü etken zamansal ve uzamsal öndörü algoritmalarınn performans farkı nedeniyle (kodlanması gereken öngörü hatasının büyümesiyle) dolaylı bir etkisi vardır. Çerçeveler arası görünürlülük değişiminin az olması varsayımımızdan yola çıkarak, ilk maddede açıklanan gerekli bit sayısının düşme miktarının, ikinci ve üçüncü maddelerde açıklanan bit sayısı artışından çok daha büyük olması beklenir.

Açıklanan işlemleri gerçekleştirmek için sıkıştırmayı yapan gönderici ve göz dizisini yeniden oluşturan alıcı için aşağıdaki adımlar tanımlanmıştır.

Gönderici:

1. Görünülürlüğün saptanması
2. Kodlanacak bölgeleri tanımlama
3. Bölgedeki yüzleri ifade etmek için tek tek ziyaret etme
4. Entropi tabanlı kodlama
5. Öngörü Yöntemleri
6. Sayısal ifade yöntemi (nicelleştirne)
7. Entropi tabanlı kodlama
8. Alıcının uygulayacağı adımları gerçekleştirme
   8.1. Ters nicelleştirme işlemi
   8.2. Öngörü Yöntemleri
   8.3. Alıcının aldığı bilgiyi oluşturma

Gönderici birinci adımdan dördüncü adıma kadar, çerçeveler arasındaki görünür alanda oluşan fark bölgeleri kodlamaktadır. Daha sonraki üç adım görünür düğüm noktalarınn kodlanmasını içerir. Son olarak sekizinci adımda alıcı tarafın aldığı bilgilerle oluşturacağı göz dizisini oluşturarak, bir sonraki çerçevede uygulanacak zamansal öngörünün ilişkilendirildiği düğüm bilgilerinin aynı olması sağlanmaktadır.

Görünürlülüğün saptanması adımında bakış noktasından düğümlere ışınlar gönderilir. Bu ışınlar ilgili düğümden daha yakında herhangi bir yüzle kesiştiği takdirde, ilgili düğüm bakış noktasından görünmez şeklinde işaretlenir. Bu işlem her düğüm için yapıldığında görünürlülük tamamıyla belirlenmiş olur.

Kodlanacak bölgeleri tanımlama ve bu bölgelerdeki yüzleri tek tek ziyaret etme işleminde, önceki adımda görünürlülüğü test edilen yüzlerden hangileri ile ilgili bilgi gönderileceği bir önceki çerçeveye bakılarak karar verilir. Alıcıya bildirilmesi gereken yüzler bir önceki çerçeveden sıkıştırılan çerçeveye geçişte, görünürken

görünmez olan yada görünmezken görünür olan bölgelerdir. Bu bölgeler Edgebreaker adlı algoritmanın değiştirilmesiyle oluşturduğumuz algoritma ile işaretler haline getirilir.

Yukarıda adım dörtde belirtilen entropi tabanlı kodlama işleminde, önceki adımda oluşturulan işaretler aritmetik kodlama ile kodlanarak her işaretin karşılaşılma olasılığı yardımıyla kodlama için gerekli bit miktarı azaltılır.

Uzamsal öngörü için paralelkenar oluşturma yöntemi kullanılır. Bu yöntem kodlanan düğümün, komşu üçgen ile bir paralelkenar oluşturduğu varsayımıyla ilgili düğümün gerçek konumuna yaklaşmayı hedefler. Zamansal öngörü yöntemi olarak önceki çerçeve ile ilişkilendirilmiş ortalama algoritması kullanılır. Bu algoritma konumu tahmin edilmek istenen düğümün komşularının, önceki çerçeve ve şu anki çerçevedeki konumlarının ortalaması alır. Önceki çerçevedeki gerçek konumu ile önceki çerçevedeki ortalamasının farkına, şuanki çerçevedeki ortalamayı ekleyerek gerçek konuma yaklaşmaya çalışır. Bir başka deyişle ortalama yöntemi ile önceki çerçevede elde edilen hatanın şu anki çerçevede elde edilecek hataya çok yakın olduğunu varsayarak bu özelliği kullanır. Öngörü yönteminin sonucu ile gerçek konum arasındaki farkin alıcı tarafına gönderilmesi gerekmektedir.

Nicelleştirme adımında, devamlı gerçek sayı olan değerler, sınırlı sayıda tamsayıya çevirilir. Bu işlem belirlenen Δ genişliğindeki tüm değerlerin tek bir tamsayı ile ifade edilmesi ile sağlanır. Bu adımda Δ sayısının büyüklüğü, alıcının aldığı değerdeki geri kazanılamayacak olan hata miktarını belirler. Bu adımın sonunda sınırlı bir tam sayı kümesi oluşur ve bu sayılar entropi tabanlı yöntemle kodlanır.

Göndericinin son adımı ise, alıcının kayıplı very ile yapacağı işlemlerin aynısını yaparak, daha sonraki çerçevelerdeki öngörü hesaplarında kullanılacak olan düğüm bilgilerinin gönderici-alıcı taraflarında tamamen aynı olamsını sağlamaktadır.

Alıcı:

1. Değişim bölgelerinin alınması
   1.1. Entropi tabalı kodlama
   1.2. Alınan işaret ile ilgili yüzü ziyaret etme
   1.3. Bölgeleri tanımlama
   1.4. Görünür bölgeyi oluşturma
2. Düğüm bilgilerinin alınması
   2.1. Entropi tabanlı kodlama
   2.2. Nicelleştirme işlemini ters çevirme
   2.3. Öngörü methodları
3. Çerçevenin alıcı tarafında yeniden oluşturulması

Alıcı kısmında ilk işlem, göndericitarafından gönderilen görünür bölgenin değişimi ifade eden işaretleri almadır. Bu alınan işaretler entropi tabanlı kodlama yardımıyla, gönderici tarafında entropi kodlamadan önceki anlamlı işaretlere çevirilir. Bu işaretler ilgili bölgenin yüzlerini bir-bir ifade ederek tanımlar. Bu adım sonunda görünür olan ve olamyan bölgeler tamamıyla belirlenmiş olur.

Düğüm bilgilerin alınması ile başlayan işlem, entropi kodlayıcının yardımıyla, gönderici tarafında entropi kodlamadan önceki anlamlı tamsayılara çevirilir. Bu tamsayılar nicelleştirme işlemi ile belirlenmiş bir Δ aralığındaki tüm sayıları temsil ettiğinden, gönderilen değer bu aralığın orta değeri olarak kabul edilir. Bu değer göndericinin öngörü algoritması ile yaptığı yaklaşımın hatası olduğundan, aynı işlemler alıcı tarafında yapılarak, göndericinin (şimdi de alıcının) hesapladığı değer

ile toplanır. Bu işlem sonucunda Δ aralığının büyüklüğüne bağlı bir hata payı ile alıcı tarafında düğüm konumu belirlenmiş olur.

Yukarıda anlatılan önerdiğimiz sistemin tüm adımları gerçekleştirildiğinde, bir göz dizisinin düğümlerinden sadece görünenlerin, ve düğümler arası bağlılık bilgisinin (düğümlerin üçgen şeklinde ilişkilendirilmesi ile oluşan yüzler) de sadece görünür bölgedeki değişimi ifade edecek kadarını gönderilmesi sağlanmış olur.

Önerilen sistem, bakış noktası bağımsız olan, entropi kodlama, nicelleştirme ve öngörü hesapları için aynı yöntemleri kullanan diğer yöntemle karşılaştırılmıştır. Bu karşılaştırma "chicken crossing" adlı 3030 düğüm noktası, 5664 yüzü (üçgen yüzey) ve 400 çerçevesi olan bir dinamik göz dizisi ile yapılmıştır. Karşılaştırmada farklı üç bakış noktasına gore hata miktari – bit miktarı bilgileri bakış noktası bağımsız yöntemle karşılaştırılmıştır. Her bir düğümü ifade ermek için 15 bit harcandığı durumla 3 bit harcandığı durum aralığında birçok değer belirlenmiş ve bu değerlerde, %25 ile %47 arasında sıkıştırma kazanımı olduğu gösterilmiştir.

Aynı dinamik göz dizisi için çalışmanın başında yapılan düğümlerin yaklaşık yarısının görünmeyeceği varsayımı test edilmiştir. Bu test sonucunda üç farklı bakış noktası için, bir çerçevede bilgisi kodlanan düğümlerin ortalama sayısı 1598.6 bulunmuştur. Yine aynı dinamik göz dizisi ve üç farklı bakış noktası için görünür bölgeye giren yüzleri beilrmek için göz dizisindeki çerçeve başına 54.5 bit, görünür bölgeden çıkan yüzleri belirtmek için çerçeve başına 51.5 bit harcanmıştır. Bu bölgeleri belirtmek için harcanan bit miktarı düğüm başına hesaplandığında, sırasıyla 0.018, ve 0.017 bulunmakta ve düğüm başına harcanan toplam bit sayısının 3 ile 15 arasında değiştiği dikkate alındığında çok makul bir miktar olduğu ortaya çıkmaktadır.

Önerilen sistemin harcadığı bit miktarının sıkıştırılan göz dizisindeki nesnenin çeşitli hareketlerine ve diğer değişimlere göre değişimi bakış noktası bağımsız sıkıştırma ile karşılaştırılmıştır. Sıkıştırılan göz dizisindeki nesnenin faklı hareketlerini ve değişimlerini içeren çerçeveler incelendiğinde, önerilen yöntemin bu durumlara karşı bakış noktası bağımsız yöntem ile benzer cevap verdiği yani sıkıştırma başarılarının benzer şekilde değiştiği görülmüştür. Bu sonuç, önerilen yöntemin, bakış noktası tabanlı olmasına rağmen, hızlı hareketler ve dönme hareketleri gibi değişimlere karşı başarısız olmadığını göstermiştir.

Bu çalışmada, önerilen sistemin aynı parametrelere bağlı bakış noktası bağımsız kodlamaya gore %47'e varan sıkıştırma kazanımı olduğu ve göz dizisindeki çeşitli değişimlere karşı olumlu sonuç verdiği başarıyla gösterilmiştir. Sistemin alt adımları olan, entropi kodlama, öngörü hesaplamaları ve nicelleştirme gibi işlemleri yeni önerilecek başka methodlarla değiştirilmesi sonucu sistemin geliştirilmesinin mümkün olduğu açıktır. Görünürlülük tespiti için, fazla işlem zamanı gerektiren ama basit olan ışın-üçgen kesişimi methodu kullanılmıştır. Bu algoritmanın hızlandırılmış alternatifi ile değiştirilmesi sonucunda, önerilen sistem gerçek zamanlı bir uygulama için oldukça uygun bir yöntem olacaktır.

VIEW-DEPENDENT CODING OF ANIMATED MESH SEQUENCES

SUMMARY

Animated mesh sequence models are popularly used for many visualization of moving synthetic objects in computer simulation, film and game industries. 3D objects are represented by their surface, and all changes are represented by changes in surface.

View-dependent processing of 3D meshes are used for many applications like visualization and physical test (such as, collision, impact of objects to others and lightening). View-dependency in compression is relatively new research area.

View-dependent compression methods are represented in literature for static meshes. Moreover, some view-dependent methods for mesh sequences are represented. These methods mostly based on coarse quantization or similar methods that reduce bitrate for invisible parts.

In our study, we propose a new method that completely removes invisible vertices from coding scheme. For such a purpose, visibility for all vertices must be detected and change in visible region must sent to decoder. We use ray-triangle intersection for visibility detection. Ray triangle intersection creates rays between viewpoint and each vertex. If any face intersects ray in a point near then the tested vertex, then the vertex is flaged as invisible. Otherwise, the vertex is flagged as visible.

After detecting visibility, regions (faces), which become visible or invisible currently, must be sent to decoder. For this purpose, a revised version of Edgebreaker is developed. Resulting symbols of the algorithm are encoded with arithmetic coding. Decoder does inverse of each step reversely to recover connectivity of visible regions.

Vertices in visible region are predictied with predictors: parallelogram predictor if no temporal reference available and motion vector averaging predictor otherwise. Prediction errors are quantized with uniform deadzone quantizer and encoded with adaptive arithmetic coding. Decoder does inverse of each step reversely to recover geometry (vertex positions) of current frame.

We compare our proposed method with viewpoint independent system, which uses same predictor, quantizer and entropy coder. We used proposed method with three viewpoints to compress chicken crossing mesh. Chicken crossing mesh has 3030 vertices, 5664 faces and 400 frames.

Our experiments showed that view-dependent coding significantly reduces bitrate (%25 to %47) with compared to viewpoint independent compression.

Regions, which become visible or invisible in proceeding frames, are coded with average of 54.5 and 51.5 bits per frame respectively. This concludes their bitrates are 0.018 and 0.017 bit per vertex per frame respectively, which is significantly low with compared to bitrate required to encode geometry (3 to 15 bpvf in our test scope).

Proposed system is adaptable to most of predictive methods, quantization methods and entropy coding. Thus, proposed system could be improved by improvement in any of these steps of proposed systems.

A simple method ray triangle intersection is used. Ray triangle intersection has high computational complexity. With replacement of visibility detection method and low complexity of all other steps, proposed system is highly applicable to real-time purposes.

1. INTRODUCTION

The use of 3D meshes in computer graphics is increasing, largely because of the need of representation of real world, characters or systems. 3D animated meshes are mostly cover computer games, character animations, avatars, physical simulations, etc.

Real objects are represented by their surfaces, which are composed of connected polygons. The motion of polygons in mehses represents motion of object. Additioanlly some other structures of data like, color, materal, are used to represent objeck completely. However, we only study on shape and motion, which could be represented by connectivity and geometry of a mesh.

Polygons can vary in meshes, and we work with triangular meshes. Animated meshes are expressed by a series of meshes, and each of them is a 3D scene, which called frame. Each frame has two substructures of data: connectivity (topology) and geometry. Every corner of triangles (polygons in general case) has a 3D position. Topology (connectivity) part of the mesh handles the connectivity of vertices to create triangles. When all vertices are connected to its incident vertices as declared in topology, a 3D object surface is visually composed. Triangular faces in a sample mesh could be seean in Figure 1.1.

In other words, connectivity represents all groups of vertices that connect as triangle and creates a face. In this study we work with constant connectivity meshes, which means the connectivity of all frames are same.

Required bitrate to represent 3D animated mesh sequences are much higher than required to represent static 3D meshes. Similar to difference between video and image coding, static 3D mesh coding techniques and 3D animated mesh sequence coding techniques differs. Because of this, compression of 3D mesh sequences is very important to represent, transmitt, and store 3D objects in computer environments.

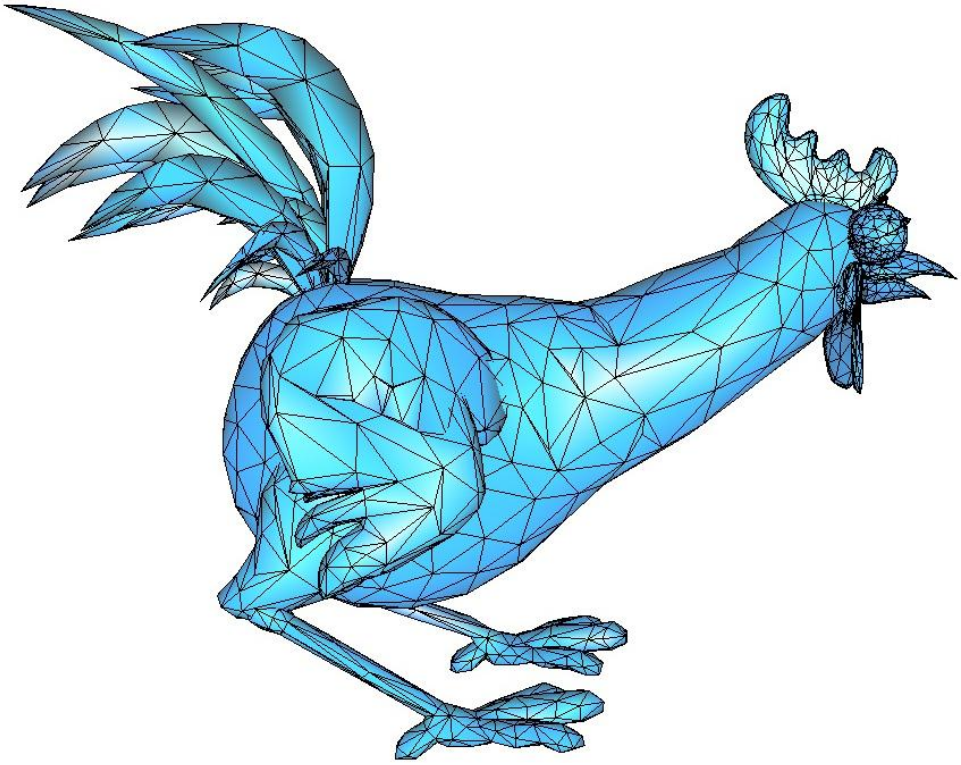


Figure 1.1: One frame from chicken crossing mesh.

View-dependency is used in many applications on 3D meshes. Most common view-dependent operations are visualization, occlusion test, effects of objects to others and lightening of object. Our purpose is to adapt view-dependancy to compression of 3D meshes.

Removing invisible vertices from coding scheme is applied for static meshes for some studies ([1], [2], [3], [4]). Additionally, some studies propose methods that allocate low bits for invisible parts of the static mesh ([5], [6], [7]). These methods reduce the number of triangles (resolution) of the invisible parts to reduce required bits for invisible parts. Such methods reduce required bits with reducing quality of invisible parts.

In our study we assume that viewpoint of the viewer is known or received by encoder side. This assumption is widely accepted by visibility-based studies. Another assumption is that connectivity of the mesh is constant over time.

The proposed method is based on removing invisible vertices in each frame of 3D mesh sequences. With a coarse prediction, nearly half of the vertices could be removed from coding scheme with removing invisible parts. When we assume that

visibility of the mesh does not change greatly over time, bitrate would significantly reduced with removing invisible parts.

Decoder must have all visible parts in each frame. In proposed system, because of removing invisible parts in each frame, change in visibility must be encoded. Vertices in region, which become visible in current frame, must be encoded with spatial reference, since there is no temporal reference is available.

In other words, there are three factors, which change bitrate of the compression:

1- Bit reduction with removing invisible vertices

2- Required bit to represent regions which become visible or invisible in current frame

3- Some vertices lie in a region that is not visible in previous frame, but becomes visible in current frame. These vertices have no temporal referene. Thus, efficiency of compression is reduced with the lack of temporal reference.

First two factors are directly effects the bitrate, while third factor is increase bitrate as much as the efficiency difference between temporal prediction and spatial prediction algorithms.

1.1 Purpose of Thesis

A view-dependent coding of 3D mesh is the coding of semi-regular meshes composed by removing invisible parts. We introduce a new system to compress animated meshes based on a viewpoint. Purpose of the study is to reduce bitrate with removing invisible parts and encoding only visible vertices. For such a purpose, visibility of the vertices must be detected and decoder must be informed about visibility of vertices.

When working with animated meshes based on viewpoint, each frame has divided into multiple regions flagged as visible or invisible to the viewpoint. Our compression system handles the detection of visible and invisible regions, the changes of regions over time, and changes of geometry in time and space.

Expected benefit is as much as the bitrate required to encode invisible vertices while drawbacks in each frame is the cost of encoding the change in visibility of regions.

While we assume that nearly half of the vertices are not visible in a frame, a good algorithm to encode visibility in current frame could result great bit reduction.

1.2 Structure of Mesh Sequences

Animated meshes are composed by meshes over time, and structure of a mesh sequence is as follows:

(M_f), $f \epsilon [0,...,F-1]$ is the mesh of frame f which composed by geometry and connectivity,

(G_f), $f \epsilon [0,...,F-1]$ is the geometry of frame f,

(T_f), $f \epsilon [0,\ldots,F-1]$ is the topology(connectivity) of frame f,

$G_f = (X_f, Y_f, Z_f)^T$, geometry is composed of 3D positions of vertexes

$p_v^{\ f}$ is the vertex v of frame f and composed of 3D position (x_v, y_v, z_v)

$(M_0,M_1,...,M_{F-1})$ is an animated mesh which has F frames over time.

1.3 Hypothesis

To completely represent all visible regions, change in visibility over frames must be coded. Change in visibility could be represented by group of regions that included into or excluded into visible region. When change of visible regions over frames are relatively small, each of these regions are composed of connected faces with small number of faces.

When visible regions of the mesh changes slowly over time, regions which included into visible region in a frame have small number of vertices according to all vertices. Thus, raise of bitrate caused by compression of these vertices with spatial reference instead of spetio-temporal reference is low.

Geometry coding allocates most of the bits in all coding scheme. Connectivity coding could be represented very effectively in vearious ways with different methods. However, geometry coding means to represent three number for each vertice. Without coding invisible vertices significant bitrate reduction might be gained.

Our claim is that overhead of coding changes in visibility significantly lower than gain of excluding invisible parts. With an algorithm, which represents changes in visibility between consequtive frames, overhead of the propose method could be very low.

2. LITERATURE REVIEW

We classified literature review to three category, coding tecniques that are basic methods in mesh compression, visibility detection methods, region growing algorithms and lastly other view-dependent techniques in literature.

2.1 Coding Techniques

In this section, we represent and give details of the most promising techniques for dynamic 3D mesh compressions from literature. Techniques for dynamic 3d mesh compression are divided into four categories in [8].

2.1.1 Clustering-based approaches

In this method, meshes are divided into subparts so that motion of each part can well described by rigid transform. As a result, motion of object is defined as motion of subparts. These clusters have motion which can be represented by transform operations. Each cluster has vertices of similar motion. Resulting presentation of mesh has two parts: rigid motion (transform) parameters of each cluster and prediction errors of each vertex.

MPEG-4 / FAMC is a standardized method, which is commonly used method with 3D affine transform of clusters ([9],[10],[11],[12],[13]). FAMC describes motions of clusters with a single 3D affine transform and error between transform result and real vertex positions.

2.1.2 PCA-based representations

Although there are lots of new versions and improvements in PCA-based methods, first method proposed in [14]. Like other data types, PCA-based approaches are used in compression of 3D animated meshes. As preliminary work, global motion compensation is applied to split elastic and rigid body motion. The difference between the rigid body motion parameterization and actual geometry could be efficiently represented by a principal component analysis (PCA).

2.1.3 Spetio-temporal predictors

Prediction-based methods predicts a vertex p_v^f with a function which uses other vertices in mesh (spatial prediction) and/or same vertex and adjacent vertices in previous meshes (temporal prediction). In other words, animation sequence is processed vertex by vertex with the help of spatially and temporally local information.

$$p_v^f = pred(v, f) + r_v^f, \tag{2.1}$$

Where p_v^f is the vertex v of frame f, pred(v,f) is the corresponding prediction for vertex, and r_v^f is the difference vector between real position and predicted position. If pred(v,f) is only based on vertices in frame f, then it is called spatial prediction (2.2). If pred(v,f) has dependency of any vertex in any frame before f, it is called temporal prediction (2.3).

$$pred(v, f) = p_{v-1}^f, \tag{2.2}$$

$$pred(v, f) = r_v^{f-1}, \tag{2.3}$$

Predictive coding techniques benefit from low computation cost, while there is no support for scalable rendering and progressive transmission, which are basically supply scalability over resolution.

2.1.4 Wavelet-based approaches

According to their representation methods, wavelet-based techniques are divided into two subcategories: Re-meshing based approaches and irregular wavelet-based approaches.

The re-meshing based techniques give best results for static 3D mesh encoding. This approach re-samples the mesh surface to obtain a semi-regular or regular topology well suited to wavelet reconstruction.

Irregular wavelet-based techniques define wavelet for mesh and encode the mesh with wavelet parameters, which change from frame to frame.

This method has low computation complexity, but the main problem is deciding wavelet filter coefficients for first frame. This situation causes the compression

inefficiency for animated meshes, which has no parametrically coherent mesh sequences.

Figure 2.1 summarizes the advantage, disadvantage, and properties of the families of 3D mesh compression techniques [8]. For this comparison, some samples from most promising approaches of families are used as listed below.

- The irregular wavelet-based encoding scheme AWC[15],
- The PCA-based approaches PCA[14], LPCA[16], and CPCA[17],
- The vertex prediction-based encoders AFX-IC[18] and Dynapack[19],
- The clustering-based encoders RT[20] and D3DMC[21].

From Figure 2.1 and explanations in this chapter, low computational cost and widely applicability occurs for predictive coding techniques in all family of approaches.

Approach		Principle	Encoding computation complexity	Progressive transmission	Scalable rendering	Applicability
Vertex-based predictive approaches	**MPEG-4 IC**	Local spatio-temporal prediction	*	no	no	all meshes
	Dynapack	Local spatio-temporal prediction	*	no	no	manifold meshes
Clustering-based approaches		Spatial segmentation, parametric motion models and temporal prediction	**	no	no	all meshes
Wavelet-based approaches	**Re-meshing**	Regular wavelet transform	***	yes	yes	manifold meshes admitting low-distortion parameterizations
	Irregular wavelets	Irregular and anisotropic wavelets transform on the top of a progressive mesh hierarchy	*	yes	yes	parametrically coherent manifold mesh sequences
PCA-based approaches		SVD decomposition performed on the set of all the frames	***	yes	no	all meshes

Figure 2.1 : Families of 3D mesh sequence coding techniques [8].

2.2 Visibility Detection Methods

The structure of a mesh is a set of triangles. For front face detection, two sub steps are applied to mesh as proposed in [4]: first is view-frustum test, which test the part of the 3D scene falls inside of the screen, and second a surface-orientation test that test the front faces, back faces and occluded faces according to the user's point of view.

The view-frustum finds which vertices are inside the view pyramid with rectangular base, which corresponds to screen. Sides of pyramids are defined as

$$a_j x + b_j y + c_j z + d_j = 0, \tag{2.4}$$

for j=1,2,3,4.

A vertex v_i is in the frustum if

$$a_j v_i^x + b_j v_i^y + c_j v_i^z + d_j > 0, \tag{2.5}$$

for j=1,2,3,4.

In order to detect which surfaces (group of triangles) are facing to the viewer, there exist two basic approaches: Face normal and ray/triangle intersection.

Face normal method based on calculating normal for each triangle, which represents a plane. Each triangle gives two normal as inner normal vector and outer normal vector. The angle between outer normal of a triangle and point of view simply gives if the triangle faces to the user. Let the calculated angle be θ_i for v_i. Then, triangle is on frontface according to the viewpoint if θ_i is between -90 and 90 degrees. Otherwise, triangle is on back face of the object (Figure 2.2 and 2.3).

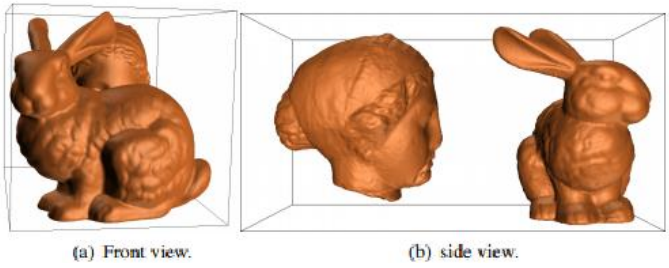


Figure 2.2 : Sample 3D scene from point of view and side view [4].

Figure 2.3 : Visibility detection by face normal [4].

Ray/triangle intersection [22] method determines the visible parts. The principle is to test the intersection between observation line (lines started with point of view and directed to the object) and triangles. We test intersected triangles with each ray. For each ray first intersection of a triangle represents the visible triangle which occludes others in the case of multiple intersections (Figure 2.4).

Figure 2.4 : Ray-triangle intersection test [4].

2.3 Region Growing Algorithms

Region growing algorithms start with an arbitrary face or edge and grow this region with neighboring faces. General purposes of region growing (or traversal) algorithms in 3D meshes are

- to encode connectivity face by face and
- to keep same order of coding for vertices in both sides (encoder-decoder).

Connectivity coding is applied for first frame of animated meshes with constant connectivity. Following frames has constant connectivity that is known by both encoder and decoder. After first frame region growing algorithms run both encoder and decoder to synchronize traversal of vertices.

Region growing algorithms mostly start with an initial face/edge and represent next face/edge with a symbol. These steps are valid for Edgebreaker ([23],[24],[25]) and some ather traversal algorithms ([26],[27]), while fifo-based method ([28]) do not require symbols to represent next face/edge.

2.3.1 FIFO-based method

As a traversal algorithm, FIFO based region-growing algorithm represented in [28]. The algorithm initiates with a seed triangle and its three edges. A region grows by

including adjacent triangle to the next edge in the queue. Algorithm guarantees that each edge in stack has two incident triangles and one of them always lies in the growing region. This algorithm based on following algorithm:

1- Arbitrarily choose initial triangle
2- Use PCM-encoding for first triangle
3- Put edges into FIFO
4- Repeat
    a. Remove first edge e_j from FIFO
    b. Find adjacent unvisited triangle t_i
    c. If t_i has unvisited vertex encode the vertex
    d. Else remove other edges of t_i from FIFO
    e. Put outer edges into FIFO

Until all triangles processed

An example run of the algorithm is given in Figure 2.5.

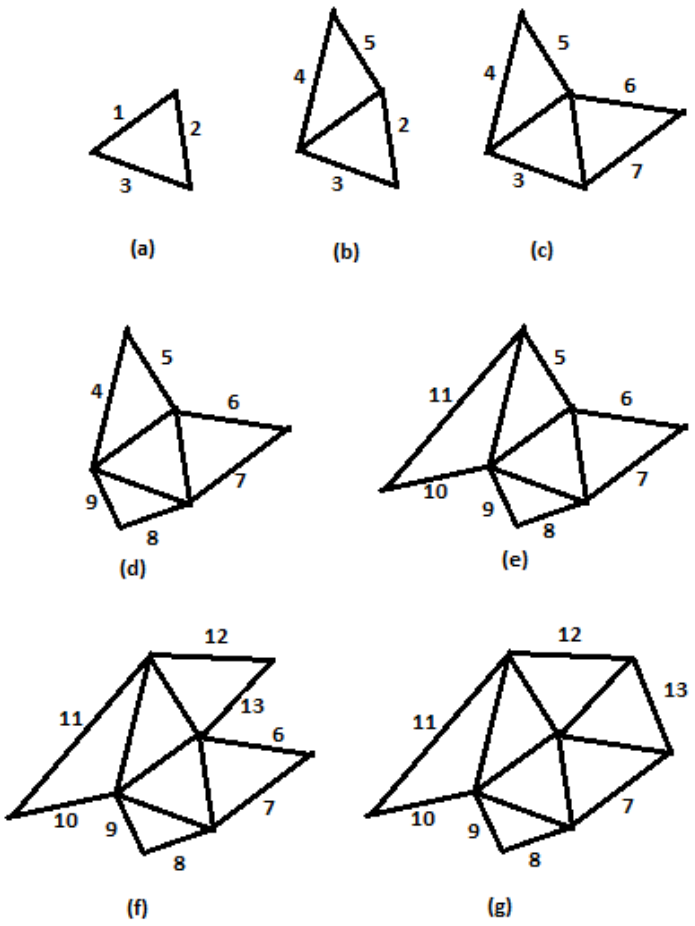


Figure 2.5 : Step by step region growing FIFO algorithm.

In Figure 2.5, Numbers over edges represent the enqueueing order of edges into FIFO, and smallest is to next to dequeue including adjacent triangle to the next edge in the queue. In Figure 2.5, (a) is a sample for step 1 to 3, (b), (c), (d), (e) and (f) are examples for step 4-c, and (g) is an example of step 4-d.

As a further explanation, Step 4-c and Stem 4-d is the growing steps of the algorithm. If next triangle is not visited yet, then Step 4-c encodes vertex position and assigns the traverse order of the vertex as the index of the vertex. Additionallyi Step 4-c assigns traverse order of face as index of face. In execution of Step 4-d, there is no newly encountered vertex. However, the newly encountered face is indexed according to traverse order of the face and the third vertex of the face must indicated to the decoder.

Fifo-based method rearranges indicies of vertices. Order of vertex occurance in traversal is accepted as vertex index in both encoder and decoder. Similarly, order of face occurances in traversal is accepted as face index. As a result of indexing faces and vertices with mutually known information, no symbol is required to represent next face of traversal. However, index of the third vertex must be sent to decoder in Step 4-d.

Fifo-based method does not require encoding symbols for next face to visit. However, as a drawback Fifo-based method reorder connectivity information and vertex indicies according to traversal order.

2.3.2 Edgebreaker

A simple method for compressing the connectivity: Edgebreaker algorithm represented in [23]. Some improvements and implementation methods are represented in [24] and [25]. Edgebreaker algorithm starts with a gate (initial edge) and defines a coding scheme to define which face is next to include in growing region. The algorithm defines five symbols as center (C), left (L), right (R), split (S) and end (E) according to ceses, which visually defined in Figure 2.6.

As in the study [23], Edgebreaker algorithm with half edge structure is as follows:

IF v is not on border THEN case C
ELSE IF v follows g
THEN IF v precedes g THEN case E ELSE case R
ELSE IF v precedes g THEN case L ELSE case S

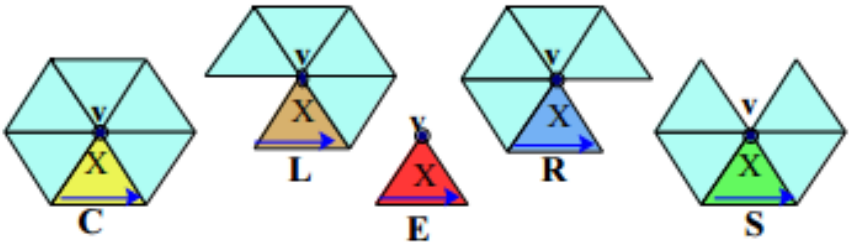


Figure 2.6 : Decision of symbols in different cases [23].

Edgebreaker algorithm requires less than 2n (guaranteed higher bound) bits and averagely 1.7 bits (expected) to encode connectivity of a mesh with n triangles.

Example traversal of Edgebreaker algorithm is given in Figure 2.7. Resulting string of Edgebreaker is "CCRRRSLCRSERRELCRRRCRRRE" for the region displayed in Figure 2.7.

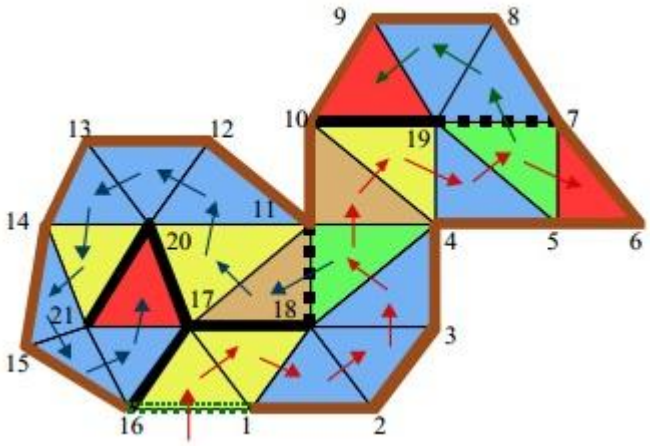


Figure 2.7 : Example traversal of Edgebreaker [23]

2.4 View-Dependent Methods in Literature

View-dependent compression techniques in literature is mostly proposed for static 3D meshes and could be divided into two category: removal of invisible parts, and coarse representation of invisible parts.

2.4.1 Removal of invisible parts

Proposed methods for removal of invisible parts are composed of two steps, first is to detect visibility, second step is to compress visible parts.

Proposes method in [3] uses a hierarchical face clustering approach. This approach initiates with number of segments equals to number of triangles. Iteratively, some

neighbouring segments are merged to compose bigger segments. The study propose a merging cost to arrenge merging operations. Each segment is encoded separately. Each segment is weighted with average normal vectors. The proposed method calculates effective area for a given viewpoint with considering weights and compression cost. Optimal number of segments are decided with compression size and transmission size. After all the steps, decided segments are encoded, which leads to reduction of required bits with the help of representing coarse topology of low weighted segments (mostly based on angle between face normal and viewing direction).

Proposed method in [4] removes invisible vertices from coding scheme and uses distance based bit allocation. As a visiblity test algorithm ray-triangle intersection method is used. Distance based bit allocation is successed by a quantization function of the distance between the objects and the viewpoint.

2.4.2 Coarse representation of invisible parts

Coarse representation of invisible partws usually applied by two main steps, first to detect visibility, second step is to compress all mesh with low bit allocation for invisible parts.

Proposed method in [6] represents a coarse representation of invisible part. A 3D mesh split into partitions. Each partition is simplified independent from other partitions. Partitioning creates a coarse mesh, which is base for refinement. Refinement of the mesh is succesed by adding vertices into coarse mesh model. The method proposes to give high priorities to the visible partitions. The progressive compression reduces required bitrate for invisible partitions in low bandwidth cases.

Proposed method in [7], is composed by three step: remeshing, scaling and zerotree-like wavelet coder. Number of nodes in tree is equals to number of edges in the mesh. The method converts original mesh into a semi-regular mesh with subdivision connectivity. Thus, resulting mesh is composed of base mesh and refinements of subdivisions. Each refinement step represents a segment (a branch in the edge-based tree). Each edge-based tree is encoded with wavelet-based progressive mesh coder separately. The Proposed method assigns weights to subdivision according to their visibility. With such a system, proposed method delivers base mesh topology

loselessly, while rest of bandwidth is uses for encoding of more important trees (refinements of more importatnt regions).

3. PROPOSED SYSTEM

3.1 Proposed Codec Steps

Our coding system is represented in Figure 3.1. Two kinds of data are sent from encoder to decoder, which represent region description (faces) and geometry (vertex positions).

Encoder is composed of twelve steps. Step-1 is a simple reading of a mesh. Steps 2-3 decide which regions are visible and invisible, and do the required region definitions to represent changes in visibility of faces/triangles. Step-4 (region traverse) is actuated by a improved version of Edgebreaker. Purpose of this step is to represent regions with encodable symbols and reduce required bitrate. In step-5, resulting symbols of region growing algorithm is entropy coded with arithmetic coding. Step-6 two kind of prediction is used. Vertices that are not visible in previous frame are encoded with spatial prediction (parallelogram prediction). Vertices that have temporal reference (which are visible in previous frame) are predicted with motion compensated averaging predictor. Prediction errors are quantized in step-7 and entropy coded in step-8. As quantization used in our system, original data in encoder and reconstructed data in decoder are differs. Step 9-12 simulates actions in decoder to keep track of values in decoder. These values are the reference of the prediction of vertices in the next frame. Simulating decoder actions in encoder guarantees that both side compute same prediction value with same reference values.

Decoder is composed by nine steps. Entropy coded symbols are received and decoded in step 1. Region traverse with received symbols are executed in step-2. Step 3 and 4 describes visible and invisible regions. Entropy encoded prediction errors are received and decoded step-5. Prediction errors are dequantized in step-6. Step 7 and 8 compute same predictions with encoder, and reconstruct vertex positions with received prediction errors. As a result, decoder has the reconstructed mesh.

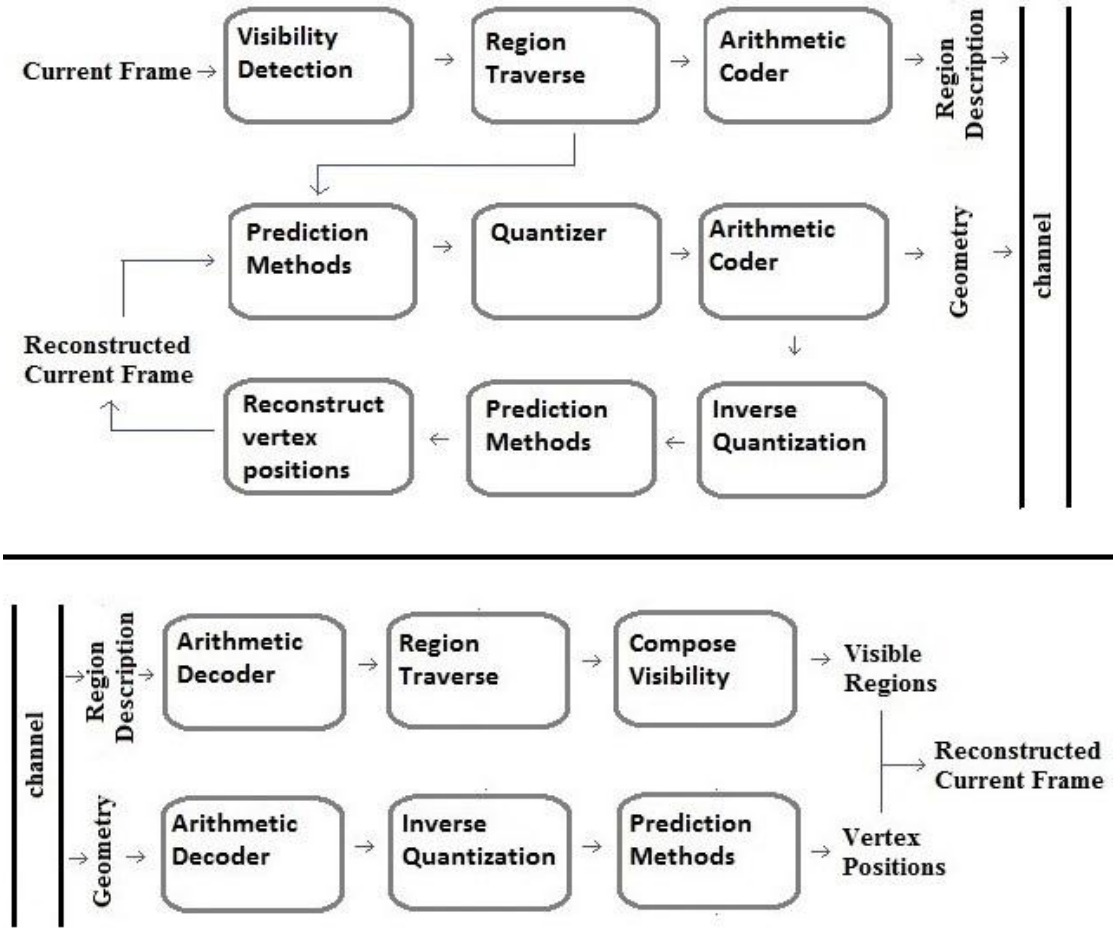


Figure 3.1 : Encoder - Decoder system for the proposed method.

3.2 Visibility Detection and Region Definitions

Ray-triangle intersection [22] is applied to meshes to decide which vertex/face visible in current frame. Suppose R^f_0 and R^f_1 represent invisible and visible regions (connected or unconnected group of faces) respectively in frame f. For any frame f we must define R^f_0 and R^f_1 to fully represent visibility. Because mesh sequences are composed by removing invisible parts, each frame has four type of region based on changes of visibility over frames:

$R^f_{(0,0)}$: invisible in both previous and current frame

$R^f_{(0,1)}$: visible parts which are invisible in previous frame

$R^f_{(1,0)}$: invisible parts which are visible in previous frame

$R^f_{(1,1)}$: visible parts which are also visible in previous frame

For first frame (mesh) M_1, suppose that we have a previous frame M_0, which has no visible face. Thus all regions in first frame are grouped into $R^1_{(0,1)}$ and $R^1_{(0,0)}$, and we could define $R^f_{(x,y)}$ for any frame f, where x =0,1 , and y=0,1. Each frame is divided into two regions as visible or invisible and four regions according to changes in visibility (Figure 3.2):

$$M_f = R_0^f + R_1^f, \tag{3.1}$$

$$M_{f-1} = R_0^{f-1} + R_1^{f-1}, \tag{3.2}$$

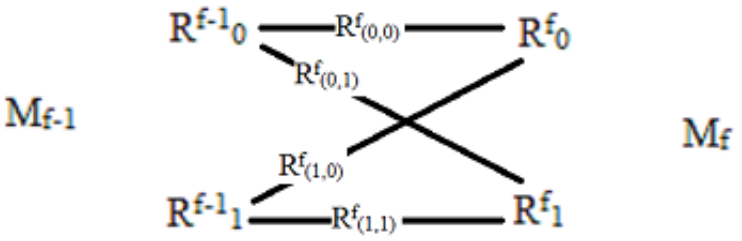


Figure 3.2 : Region definitions for transaction from frame (f-1) to f.

In coding phase of any frame f, encoder and decoder has the information $R^{f-1}{}_0$ and $R^{f-1}{}_1$. As a result of knowledge about previous frame, defining one of $R^f_{(0,0)}$ and $R^f_{(0,1)}$, also defines the other region while their union region ($R^{f-1}{}_0$) is known and there is no intersection of them. Same operations guarantee one of $R^f_{(1,0)}$ and $R^f_{(1,1)}$ and knowledge of previous frame is sufficient to define the other. In order to fully define R^f_0 and R^f_1 (so the current frame), decoder must get at least one of $R^f_{(0,0)}$, $R^f_{(0,1)}$, and at least one of $R^f_{(1,0)}$, $R^f_{(1,1)}$.

Regions, which remain visible or invisible in proceeding frames, are greatly larger than areas, which become visible or invisible in proceeding frames. As a result, we send information which defines $R^f_{(0,1)}$ and $R^f_{(1,0)}$ to the decoder.

3.3 Region Description Coding

In order to define any region R, we have developed a new version of Edgebreaker. In our scheme, encoded regions are the regions which become visible ($R^f_{(0,1)}$) or become invisible ($R^f_{(1,0)}$).

The most important reason to develop a traversal algorithm to represent regions is the varying size of regions. $R^f_{(0,1)}$ and $R^f_{(1,0)}$ are mostly composed of very small unconnected areas with a few faces. As a result, overhead to represent each unconnected region must be low. The traversal algorithm that we propose, has overhead of indicating initial face for each unconnected region, which is an obligatory information for all of alternative methods.

Edgebreaker uses an edge to represent gate g. However, we indicate the starting face of the strip and algorithm uses first edge of the indicated face as gate g, in both encoder and decoder side. With this approach, we ensure the start (overhead) of each unconnected region is represented by indicating face number.

Another reason to choose Edgebreaker and enhancing the algorithm is the low bit requirement for representing a face by using only local connectivity. In our proposed system, we encode regions, $R^f_{(0,1)}$ and $R^f_{(1,0)}$. With such small regions, Edgebreaker does not require any connectivity or geometry information of far faces. Edgerbreaker works with only adjacent faces, what makes the algorithm appropriate for encoding with local connectivity information.

Additionally, Edgebreaker represents faces with very low bitrate. Since we encode a region R, chain coding for bounding edges could be used to represent a region. However, chain coding requires high bitrates for representing each edge in bounding edges. As an example, assume average edge number of a vertex is seven (there are six possibility for next edge in a chain of edges), which is very acceptable for general meshes. Required bitrate would be $\log_2(7-1) = 2.58,$ bits per edge for every edge in boundary. Additionally, it is hard to fit any probabilistic model to represent next edge in chain, while there is no meaningfull relation between edges. Morever, for small regions, representing bounding edges, instead of included faces, requires the representation of more number of symbols with more bits for each.

Because of the benefits of traversal algorithms and benefits of Edgebreaker itself, Edgebreaker is adopted for our proposed method.

We define symbols split (S), right (R), left (L) and end (E). In our coding scheme, R replaces C symbol in Edgebreaker. In our preliminary test, we observe that replacing C with L or R changes the probability distributions and so entropy coding efficiency

insignificantly. Starting with a gate g and a triangle X, coding decisions in possible cases are shown in Figure 3.3.

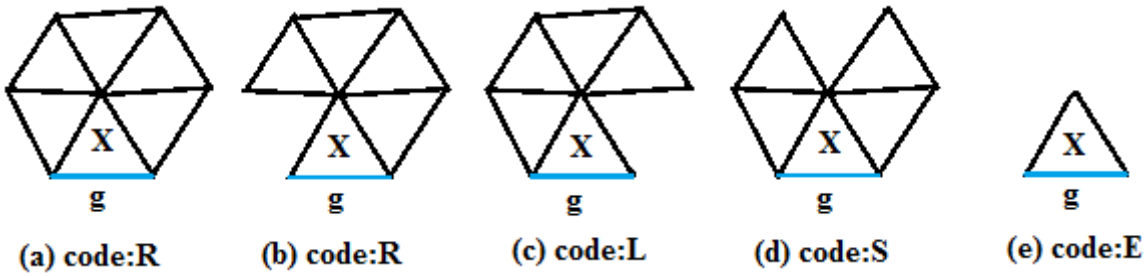


Figure 3.3 : Decisions in different cases of boundary.

In Figure 3.3, (a) case is R that replaces case C in Edgebreaker, (b), (c), (d) and (e) are same cases with Edgebreaker. Furthermore, we define a possibility space for decisions with the help of our region definitions in our system. In first step, we reduce number of possible symbols with the help of boundaries that are known by decoder. We encode two group of regions which are subset of $R^{f}_{(0,1)}$ or $R^{f}_{(1,0)}$.

Any face lies in R^{f-1}_{0} (invisible in previous frame) could not be included into $R^{f}_{(1,0)}$, which represents regions invisible in current frame, but visible in previous frame. Likely, faces in R^{f-1}_{1} are excluded with faces in $R^{f}_{(0,1)}$.

Since the purpose of the traversal algorithm is to visit all faces of a region for once, any visited face could not be the next to visit.

In case of $R^{f}_{(1,0)}$, any edge in

1- R^{f-1}_{0}
2- Already visited regions
3- Boundary of $R^{f}_{(1,0)}$

is boundary, and first two types of boundaries are known by decoder.

In case of coding $R^{f}_{(0,1)}$, any edge in

1- R^{f-1}_{1}
2- Already visited regions (including $R^{f}_{(1,0)}$)
3- Boundary of $R^{f}_{(0,1)}$

is boundary, and first two types of boundaries are known by decoder.

In order to reduce possible symbols required to encode next face, we exploit the border knowledge in decoder. According to borders around gate that is known by decoder there are four different cases (Figure 3.4).

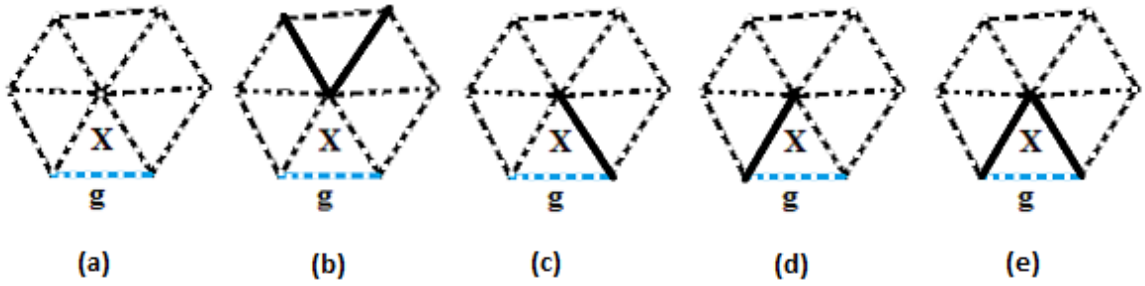


Figure 3.4 : Different cases of borders known at decoder.

In Figure 3.4, bold edges represent the borders known by decoder. Blue edge is the gate g, and other dashed lines are unknown to decoder or irrelevant to the decision.

With no payload and nearly no computational cost, for our specific purpose of coding discrete subregions of a mesh, required bits to encode region description is significantly reduced. For (a) and (b) cases, possible symbols are S, L, R and E (LERS-model), for (c) case possible symbols are L and E (LE-model), for (d) case possible symbols are R and E (RE-model), for (e) case possible symbol is E (E-model). Decision for E-model is known by decoder and no encoding required. With the help of this structure, entropy coding use probabilities symbols in current model to reduce bitrate.

In an experiment with our system on the chicken crossing mesh sequence, the following probabilities occur for Edgebreaker with four symbols (replacing C with R) in Table 3.1 and porposed modification in Table 3.2.

As a result of these probabilities, entropy of Edgebreaker with four symbols (replacing C with R) is 1.68.

Table 3.1 gives probabilities of each symbol for each probability model in proposed traversal algorithm with same mesh sequence.

As a result of the probabilities in Table 3.2, the entropy of the proposed version of the Edgebreaker is 1.62 which is lower than the Edgebreaker with 4 symbols (1.68) and absolutely lower than the original Edgebreaker (with 5 symbols).

Table 3.1 : Probabilities of symbols in Edgebreaker (with 4 symbols).

p(S)	0,073
p(L)	0,401
p(R)	0,418
p(E)	0,109

Table 3.2 : Probabilities of symbols in the proposed traversal algorithm.

	p(S)	p(L)	p(R)	p(E)
LERS-model	0,076	0,403	0,426	0,095
LE-model	0	0,651	0	0,349
RE-model	0	0	0,471	0,529
E-model	0	0	0	1

3.4 Geometry Coding

In our coding scheme we encode two kinds of vertices, which are inside of $R^{f}_{(0,1)}$ or $R^{f}_{(1,1)}$. Both regions are composed of smaller regions which are unconnected with each other. With such a case, the decoder has knowledge of spatially or temporally local information.

Vertices inside $R^{f}_{(0,1)}$ have local spatial reference, but have no temporal reference. Vertices inside $R^{f}_{(1,1)}$ have temporal and spatial reference for previous frame (f-1) and spatial reference for current frame (f), but have no guarantee of higher temporal reference (…,f-3,f-2). In this case, geometry compression methods that require temporally or spatially global information (rigid body motion translation, PCA, Wavelet-base methods, etc..) are not appropriate for the proposed scheme.

Consequently, we use a prediction based geometry compression technique. Advantages of the prediction based techniques are

1- Appropriate for both spatial and temporal reference

2- Local information is enough

3- Very fast (applicable to real-time)

4- Appropriate for all kind of meshes

For spatial prediction, parallelogram prediction is used. For temporal prediction, averaging prediction is used with spatial and temporal reference (Figure 3.5) formulated for a vertex v in frame f as follows:

$$pred_{mvavg}(v, f) = pred_{avg}(v, f) - pred_{avg}(v, f-1) + p_v^{f-1}, \quad \textbf{(3.3)}$$

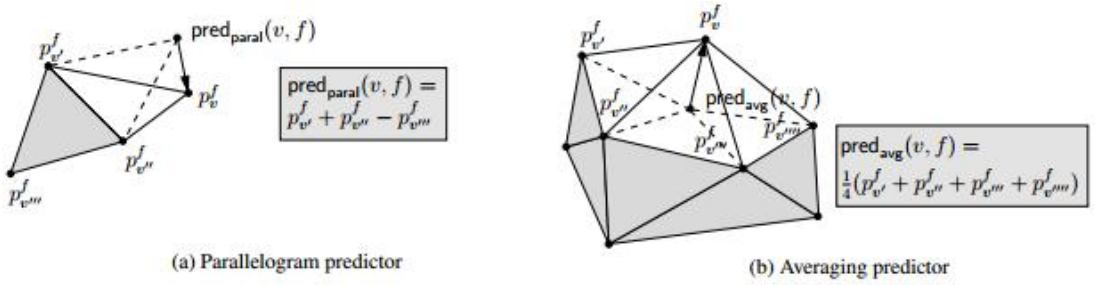


Figure 3.5 : Parallelogram and averaging prediction [28].

Here, $pred_{avg}(v, f)$ is the average of all encoded adjacent vertices to the vertex v in frame f. Reconstructed vertices are known at the decoder. As another notation

$$r_v^{f-1} = p_v^{f-1} - pred_{avg}(v, f-1), \quad \textbf{(3.4)}$$

is the error of averaging prediction in frame f-1. Prediction error in last frame (r_v^{f-1}) is the prediction of the prediction error in f.

$$pred_{mvavg}(v, f) = pred_{avg}(v, f) + r_v^{f-1}, \quad \textbf{(3.5)}$$

Parallel prediction uses only vertices connected to the p_v^f . Motion vector averaging predictor only uses vertices connected to p_v^f in the current and the previous frame.

3.5 Quantization

We quantize residuals by dead zone uniform quantizer with quantization bin width 2Δ around zero and Δ everywhere else. The value of Δ decides the amount of data loss. Graphical representation of deadzone uniform quantizer is given in Figure 3.6.

Advantage of uniform quantizer is to distribute irreversible data loss uniformly over definition space (equal-sized bins). With the help of equal sized but different

probability bins, uniform quantizer is very effective with entropy coding methods. Entropy coders exploit probabilities of bins to represent data with lower bitrates.

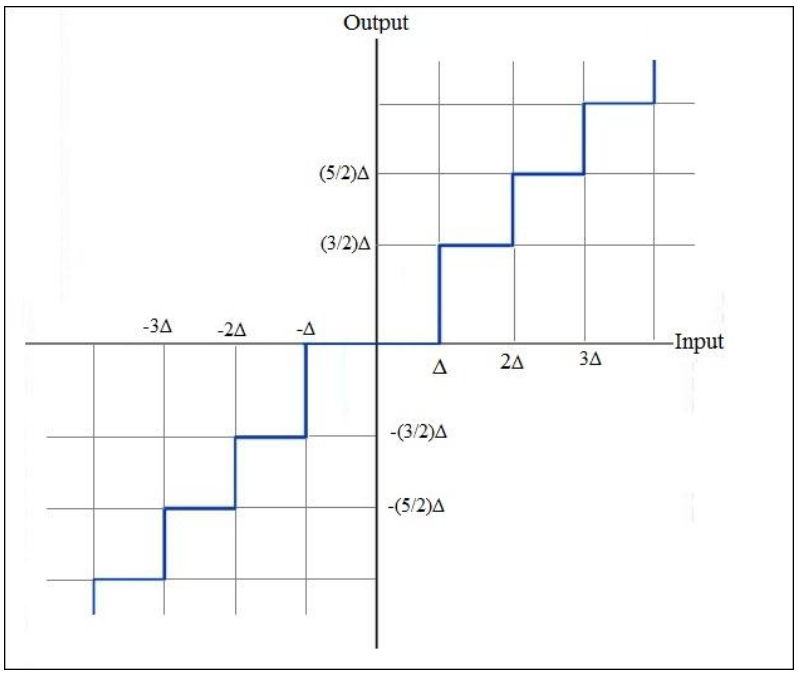


Figure 3.6 : Deadzone uniform quantizer.

3.6 Entropy Coding

Quantized residuals are encoded with order-0 adaptive arithmetic coding as an entropy coder represented in [29]. We use probabilities of all previous frames to compose probability map for current frame. Hence, bitrate allocation of residuals is dependent on all previous frames, which result with one advantage and one disadvantage.

In implementation, there is a maximum frequency which indicates maximum number of occurance of a symbol. If this maximum frequency is reached by cumulative frequencies of symbols, all frequencies are divided by two. This detail makes implementation easier by supplying a high limit to frequency values. Additionally, dividing probabilities by two cause the probabilities of later frames to have higher weight on probability model.

The advantage is that, system adapts the probabilities better after a few frames and so reduces bitrate. Because, probability model is created by more samples (with previous frames).

As a disadvantage, fast changes in prediction affects the probability distribution of residuals negatively. Fast changes are sudden changes in motion and size such as; start or end of a fast motion and likely start or end of a fast expansion, deforming, etc. These instant changes in probability effects negatively the adaptive probability space of arithmetic coding. However, if these changes are continuous, then the model adapts it after a few frames, otherwise it is not a great overhead for bitrate to allocate bits ineffectively (good but not very near to optimal) for a few frames.

4. EXPERIMENTS

In our experiments, main criteria is how much bit reduction is accomplished compared to same system with full mesh compression methods which do not use visibility detection to remove invisible parts.

In our system, if we suppose all regions are visible in all frames, then it becomes a viewpoint independent predictive compression system. For experimental purposes, we use this ability of codec. Then resulting viewpoint independent system (Figure 4.1) is used in comparisons.

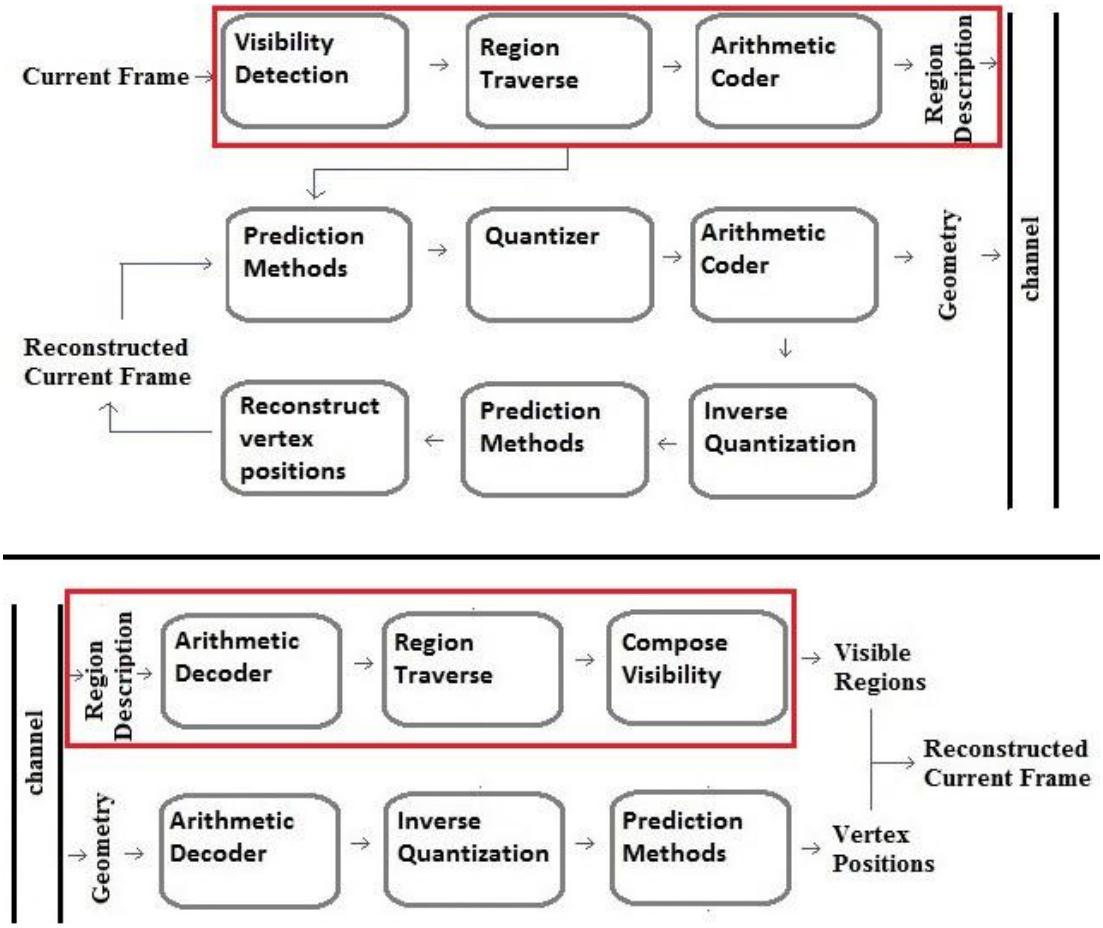


Figure 4.1 : Viewpoint independent codec schema.

Standard compression methods send connectivity at once, after that both encoder and decoder has connectivity information, which is constant. To simulate this case, first

frame is encoded with our proposed system. Following frames encoded with all steps except red area (step 2-5 of encoder and step 1-4 of decoder) in Figure 4.1. With this framework, a viewpoint independent compression system with same prediction algorithms and same quantization parameter is composed (VIC).

In Figure 4.2 VIC represents the viewpoint independent compression, VDC-1, VDC-2 and VDC-3 represent result of proposed view-dependent compression system with three different viewpoints for mesh sequence of chicken crossing. Chicken crossing mesh sequence is composed of many kind of motion form such as; steady, slow motion, fast motion, partial motion, slow turn and fast turn. Experiments on chicken crossing sequence also test behavior of methods in various motion types of object.

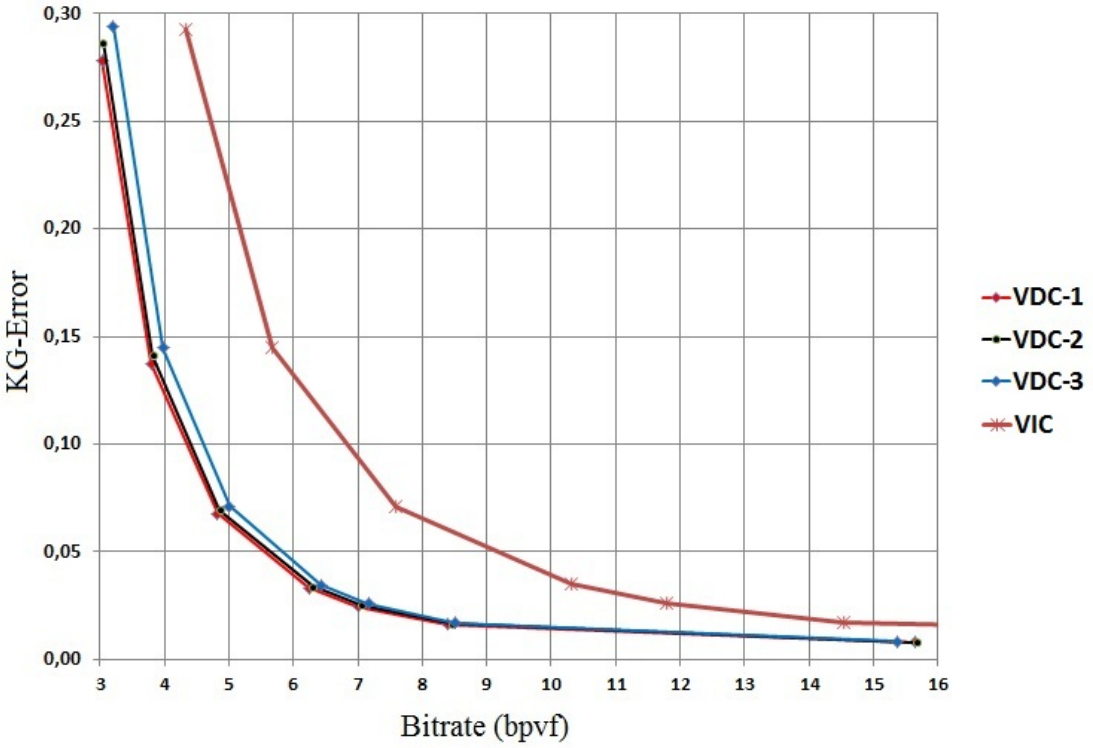


Figure 4.2 : Compression of chicken crossing sequence.

Results show that overhead of coding change in visibility are lower than the gain of bitrate reduction with removing invisible parts. According to tests in three viewpoints, invisible parts have average of 1498.4 vertices, which is not encoded. A significant compression gain (25% to 47% in our experiments) is achieved, using proposed view-dependent compression method.

R-D graph in Figure 4.2 represent both region description and geometry bits. Geometry data is composed composed by traverse of two region group $R^f_{(0,1)}$ and $R^f_{(1,0)}$. In same mesh sequence, allocated bits for these regions are as follows:

Bitrate ($R^f_{(0,1)}$) = 0.018 bpvf,

Bitrate ($R^f_{(1,0)}$) = 0.017 bpvf

and this bitrates are constant over distortion. Both of $R^f_{(0,1)}$ and $R^f_{(1,0)}$ are composed of unconnected regions, which are group of connected faces. Each connected face group represents a face strip. Average length of strips are 12.86 face/symbols (L,E,R or S) in this experiment. Rest of bits are allocated to encode position of visible vertices (geometry of $R^f_{(0,1)}$ and $R^f_{(1,1)}$). Average unconnected region number is 105.7 region per frame.

Considering that chicken crossing has 3030 vertices, 5664 faces and 400 frames, average of 1358 faces are encoded for each frame to represent changes in visibility ($R^f_{(0,1)}$ and $R^f_{(1,0)}$).

More detailed data obout the experiment is given in Tables 4.1 to 4.4. Tables 4.1 to 4.4 are composed by applying algorithms each frame of sequence with different quantization size Δ.

Table 4.1 : Detailed result of VDC-1 for chicken crosing.

	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	Δ_7
Rate	15.648	8.397	7.004	6.243	4.818	3.792	3.035
Min. Rate	8.784	3.317	2.537	2.131	1.522	1.274	1.055
Max. Rate	27.168	21.333	19.762	18.73	16.507	14.342	12.187
Distortion	0.008	0.016	0.025	0.033	0.068	0.138	0.278
Min. Dist.	0.006	0.011	0.016	0.022	0.044	0.093	0.191
Max. Dist.	0.010	0.020	0.030	0.041	0.081	0.162	0.330

Table 4.2 : Detailed result of VDC-2 for chicken crosing.

	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	Δ_7
Rate	15.662	8.462	7.064	6.300	4.859	3.813	3.040
Min. Rate	9.624	4.438	3.535	3.100	2.297	1.745	1.389
Max. Rate	24.735	18.863	17.329	16.413	14.345	12.565	10.622
Distortion	0.008	0.017	0.025	0.034	0.070	0.141	0.287
Min. Dist.	0.006	0.011	0.017	0.023	0.046	0.095	0.199
Max. Dist.	0.010	0.020	0.029	0.040	0.079	0.159	0.319

Table 4.3 : Detailed result of VDC-3 for chicken crosing.

	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	Δ_7
Rate	15.361	8.517	7.168	6.422	5.011	3.975	3.205
Min. Rate	9.428	4.403	3.319	2.829	2.135	1.700	1.410
Max. Rate	25.113	20.162	18.808	17.889	16.025	14.095	12.088
Distortion	0.008	0.017	0.026	0.035	0.071	0.145	0.294
Min. Dist.	0.006	0.011	0.016	0.021	0.044	0.091	0.190
Max. Dist.	0.010	0.021	0.032	0.042	0.088	0.179	0.364

Table 4.4 : Detailed result of VIC for chicken crosing.

	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	Δ_7
Rate	29.002	14.534	11.789	10.317	7.589	5.681	4.324
Min. Rate	19.005	7.570	5.785	4.179	3.530	2.696	1.978
Max. Rate	42.038	33.486	30.912	29.167	25.135	21.294	17.618
Distortion	0.008	0.017	0.026	0.035	0.071	0.145	0.293
Min. Dist.	0.005	0.009	0.014	0.019	0.039	0.084	0.180
Max. Dist.	0.010	0.020	0.030	0.041	0.081	0.164	0.331

Chicken crossing mesh sequence has different motions of a chicken. More steady frames of the sequence have lower bitrate, at most 68% lower than average bitrate. Fast motion frames have higer bitrate at most equal to 402% of average bitrate. Additionally these changes of bitrate over frames depend on viewpoint of VDC. Bitrate for any frame varies between

- 0.32 times of average and 4.02 times of average in VDC-1,
- 0.5 times of average and 3.49 times of average in VDC-2,
- 0.43 times of average and 3.77 times of average in VDC-3 and
- 0.46 times of average and 4.07 times of average in VIC.

Ratio of minimum bitrate of a frame and average bitrate over all frames varies between algorithms slightly, likely to ratio of maximum bitrate of a frame and average bitrate over all frames. Proposed compression (VDC-1, VDC-2, and VDC-3) and VIC behave similarly for different motion types. As a result, it is concluded that VDC is not more vulnerable to these kind of motion type than VIC.

Some example Figures from the experiment are represented in Figure 4.3 to Figure 4.6. Figure 4.3 is the picture of visible parts according to viewpoint-1 (viewpoint of VDC-1). VDC-1 reconstruction of a frame with quantization Δ_4, which averagely has 6.243 bpvf, is represented in Figure 4.4, and with quantization Δ_7, which averagely has 3.035 bpvf, is represented in Figure 4.5. Figure 4.6 represent the side view.

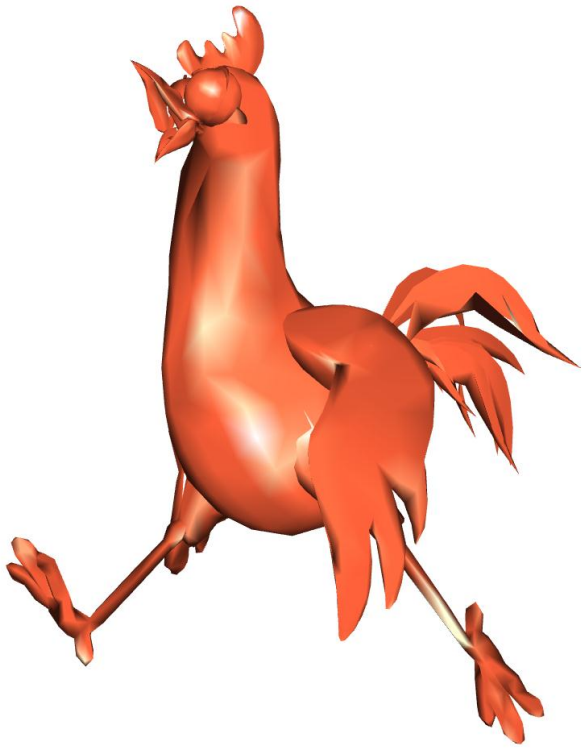


Figure 4.3 : A frame from chicken crossing with viewpoint-1

Figure 4.4 : A restored frame from chicken crossing with VDC-1 and Δ_4.

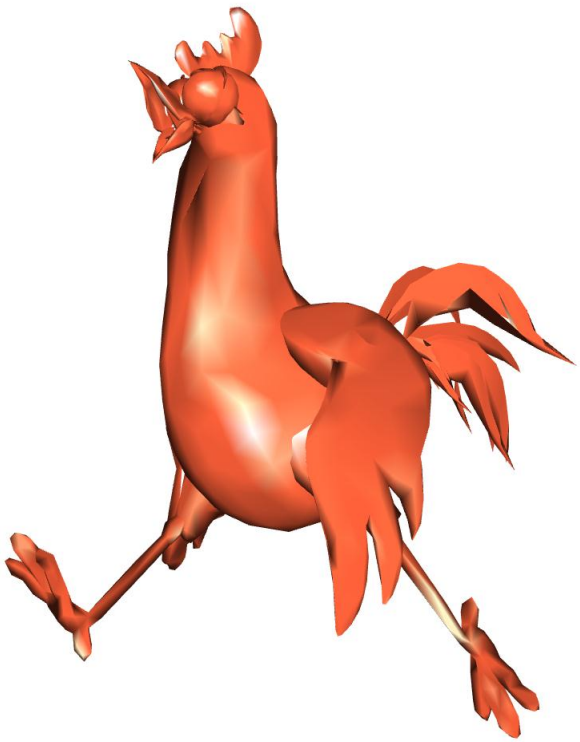


Figure 4.5 : A restored frame from chicken crossing with VDC-1 and Δ_7.

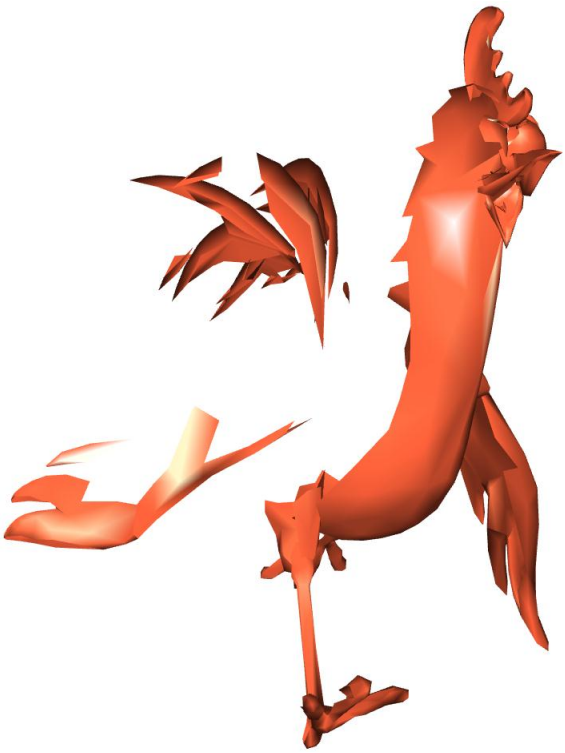


Figure 4.6 : Side view of the visible part according to viewpoint-1.

Even with this low bitrate, resulting meshes are very similar to original. VDC-1 with Δ_4 reconstruct very similar mesh to original, while VDC-1 with Δ_7 has little more variation from original mesh (an example artifact could be seen on the wing).

One problem of the proposed scheme is the special case error of ray-triangle intersection. If a face is visible while its vertices are invisible, ray-triangle intersection could not detect the visible face. Result of this special case error could be seen in the beak, eye and wing of the chicken.

5. CONCLUSION

In this study, we presented predictive compression of 3D mesh sequences with encoding only visible vertices in current frame. With the results of experiments, we showed that view-dependent compression of mesh sequences exploits visibility of mesh to reduce bitrate significantly (%25 to %47 in experiment scope).

With the high speed of linear predictive methods, proposed system also supports real-time compression.

Furthermore, our codec design is based on spetio-temporal prediction, quantization and entropy coding steps, which are easy to replace with new methods. Improvements in these substeps could be applied to proposed scheme to increase compression performance.

ACKNOWLEDGEMENTS

The chicken character was created by Andrew Glassner, Tom McClure, Scott Benza, and Mark Van Langeveld. This short sequence of connectivity and vertex position data is distributed solely for the purpose of comparison of geometry compression techniques.

REFERENCES

[1] **Yang, S., Kim, C. S. and Kuo, C. C. J.** (2002). View-Dependent Progressive mesh Coding for Graphic Streaming. Multimedia Systems and Applications IV.

[2] **Payan, F., Meriaux, F. and Antonini, M.** (2008). View-Dependent Coding of 3D Scenes. Informatique, Sıgnaux et Systemes de Sophia Antipolis Technical Report, ISRN I3S/RR-2008-17-FR

[3] **Guan, W., Cai, J., Zheng, J. and Chen, C. W.** (2008). Segmentation based View-Dependent 3D Graphics Model Transmission, IEEE Transactions on Multimedia, Volumn: 10, Issue: 5, pp. 724-734.

[4] **Payan, F., Mériaux, F. and Antonini, M.** (2009). View-dependent geometry coding of 3D Scenes, In Proceedings of IEEE International Conference in Image Processing (ICIP), Cairo, Egypt.

[5] **Alliez, P., Laurent, N., Sanson, H. and Schmitt, F.** (2003). Efficient view-dependent refinement of 3D meshes using √3-Subdivision. The Visual Computer, Volume 19, Issue 4, pp. 205-221.

[6] **Yang, S., Kim, C. S. and Kuo, C. C. J.** (2002). View-Dependent Progressive mesh Coding Based on Partitioning. Visual Communications and Image Processing.

[7] **Guan, W., Cai, J., Zhang, J. and Zheng, J.** (2010). Progressive Coding and Illumination and View Dependent Transmission of 3D Meshes Using R-D Optimization. IEEE Transactions on Circuit andSystems for Video Technology, Volume: 20, Issue:4, pp. 575-586.

[8] **Mamou, K., Zaharia, T. and Preteux, F.** (2005). A preliminary evaluation of 3D mesh animation coding techniques. In SPIE Conference on Mathmetical Methods in Pattern and Image Analysis, San Diego, USA, 44-55**.**

[9] **Mamou. K., Zaharia, T., Prêteux, F., Sefanoski, N. and Ostermann, J.** (2008) Frame-Based Compression of Animated Meshes in MPEG-4, in IEEE International Conference on Multimedia and Expo, pp. 1121-1124.

[10] **Mamou, K., Stefanoski, N., Kirchhoffer, H., Müller, K., Zaharia, T., Preteux, F., Marpe, D. and Osterman, J.** (2008). The New MPEG-4/FAMC Standard for Animated 3D Mesh Compression, 3DTV Conference, Istanbul, Turkey.

[11] **Stefanoski, N. and Ostermann, J.** (2008). Spatialy and Temporally Scalable Compression of Animated 3D Meshes with MPEG-4 / FAMC, IEEE International Conference on Image Processing, Oct 2008.

[12] **Mamou, K., Zaharia, T. and Prêteux, F.** (2008). FAMC: The MPEG-4 Standard for Animated Mesh Compression. IEEE International Conference on Image Processing, pp. 2676-2679.

[13] **Petrik, O. and Vasa, L.** (2011). Improvements of MPEG-4 Standard FAMC for Efficient 3D Animation Compression, 3DTV Conference.

[14] **Alexa, M. and Müller, W.** (2000). Representing animations by principal components, Computer Graphics Forum, Vol. 19, August, 411-8.

[15] **Guskow, I. and Khodakovskyi A.** (2005). Wavelet compression of parametrically coherent mesh sequences. ACM Siggraph Symposium on Computer Animation, July.

[16] **Karni, Z. and Gotsman, C.** (2004). Compression of soft-body animation sequences. Computer Graphic: Vol. 28. (pp. 25-34).

[17] **Sattler, M., Sarlette, R. and Klein, R.** (2005). Simple and efficient compression of animated sequences, ACM Siggraph Symposium on Computer Animation, July.

[18] **Jang, E. S., Kim, J. D. K., Jung, S. Y., Hani M. J., Wooi S. O. and Lee, S. J.** (2004). Interpolator data compression for MPEG-4 animation. In IEEE transaction on Circuits and Systems for Video Technolog, Vol. 14.

[19] **Ibarria, L. and Rossignac, J.** (2003). Dynapack: space-time compression of the 3D animations of triangle meshes with fixed connectivity, ACM Siggraph Symposium on Computer Animation, pp126-13, San Diego, California.

[20] **Collins, G. and Hilton, A.** (2005). A rigid transform basis for animation compression and level of detail, in Vision, Video and Graphics.

[21] **Müller, K., Smolic, A., Kauntzner, M., Eisert, P. and Wiegand, T**. (2004). Predictive compression of dynamic 3D meshes, ISO/IEC/JTCI/SC29/WG11/, MPEG04/M11240, Palma de Mallorca, Spain, October

[22] **Möllerand, T., and Trumbore, B.** (1997). Fast, minimum storage ray-triangle intersections. Graphics Tools.

[23] **Rossignac, J.** (1999). Edgebreaker connectivity compression for triangle meshes, IEEE Transactions of Visualization and Computer Graphics 5(1), 47-61.

[24] **Rossignac, J., Safanova, A. and Szymezak, A.** (2001). 3D Compression Made Simple: Edgebreaker on a Corner-Table, Shape Modeling International Conference, Gemoa, Italy.

[25] **Rossignac, J., Lopes, H., Safanove, A., Tavares, G. and Szymczak, A.** (2001). Edgebreaker: A Simple Compression for Surfaces with Handles, in Proceedings of the 7. ACM Symposium on Solid Modeling and Applications.

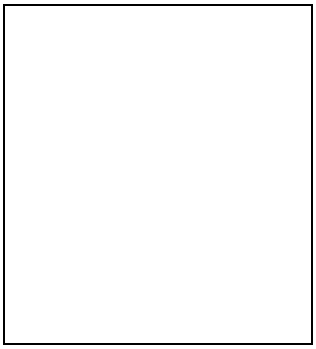
[26] **Jong, B. S., Yang, W. H., Tseng, J. L. and Lin, T. W.** (2005). An efficient connectivity compression for triangular meshes. Proc. ACIS-ICIS, pp. 583-588.

[27] **Liu, Y., Liu, X. and Wu, E.** (2007). Variable Code-Mode Based Connectivity Compression for Triangular Mehses. IEEE International Conference on Computer-Aided Design and Computer Graphics, Beijing.

[28] **Stefanoski, N. and Ostermann, J.** (2006). Connectivity-Guided Predictive Compression of Dynamic 3D Meshesl. Image Processing, IEEE International Conference.

[29] **Witten, I. H., Neal, R. M. and Cleary, J. G.** (1987). Arithmetic coding for data compression, Communications of the ACM, June.

CURRICULUM VITAE



Name Surname: Semih ÇELİK

Place and Date of Birth: Zeytinburnu, 05.01.1988

Adress: Mehmet Akif Mah. Yenice Sok. No 38. Küçükçekmece/İSTANBUL

E-Mail: celik.semih@gmail.com semcelik@itu.edu.tr

B.Sc.: Istanbul Kültür University - Computer Engineering

M.Sc.:

Professional Experience and Rewards:

1. Place of Free Style Category in 10th International METU Robotics Days (2013), with project CLOAK

1. Place of Free Style Category in ITU Olimpics of Robots 2012, with project, MIDAS

2. Place of Open – Best of Show Category in Robogames (USA, 2012), with project MIDAS

Moreover, eleven more rewards on national and international robotic competitions