ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

MSc THESIS

Ammar Abbas ELMAS

INVESTIGATION OF SINGLE-RATE TRIANGULAR 3D MESH COMPRESSION ALGORITHMS

DEPARTMENT OF COMPUTER ENGINEERING

ADANA-2019

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

INVESTIGATION OF SINGLE-RATE TRIANGULAR 3D MESH COMPRESSION ALGORITHMS

Ammar Abbas ELMAS

MSc THESIS

DEPARTMENT OF COMPUTER ENGINEERING

We certify that the thesis titled above was reviewed and approved for the award of the degree of Master of Science by the board of jury on 11/01/2019

……………………………….	…………………………………………...	………………………………………....
Assoc.Prof.Dr. Mustafa ORAL	Assist.Prof.Dr. Buse Melis ÖZYILDIRIM	Assist.Prof.Dr. Mümine KAYA KELEŞ
SUPERVISOR	MEMBER	MEMBER

This MSc Thesis is written at the Department of Institute of Natural and Applied Sciences of Çukurova University
Registration Number:

Prof. Dr. Mustafa GÖK
Director
Institute of Natural and Applied Sciences

ABSTRACT

MSc THESIS

INVESTIGATION OF SINGLE-RATE TRIANGULAR 3D MESH COMPRESSION ALGORITHMS

Ammar Abbas ELMAS

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF COMPUTER ENGINEERING

Supervisor : Assoc. Prof. Dr. Mustafa ORAL
Year: 2019, Pages: 95
Jury : Assoc. Prof. Dr. Mustafa ORAL
: Assist. Prof. Dr. Buse Melis ÖZYILDIRIM
: Assist. Prof. Dr. Mümine KAYA KELEŞ

In this thesis, currently available 3D mesh compression algorithms, frameworks, libraries etc. are investigated. Especially, the algorithms that are popular in survey papers but don't have any implementation or had outdated implementation or no published version is available, are gathered together and compiled accordingly. According to the benchmark test results, current best general-purpose data compression methods are identified and applied as the last stage of mesh compression. Results are compared in order to demonstrate the current state of single-rate 3D mesh compression performance with the current best general-purpose data compression methods.

Keywords: 3D Mesh, Mesh Compression, Single-Rate Mesh Compression, 3D Model Compression, Data Compression

ÖZ

YÜKSEK LİSANS TEZİ

STATİK ÜÇGENSEL 3 BOYUTLU ÖRGÜ SIKIŞTIRMA ALGORİTMALARININ İNCELENMESİ

Ammar Abbas ELMAS

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Danışman : Doç. Dr. Mustafa ORAL
Yıl: 2019, Sayfa: 95
Jüri : Doç. Dr. Mustafa ORAL
: Dr. Öğr. Üyesi Mümine KAYA KELEŞ
: Dr. Öğr. Üyesi Buse Melis ÖZYILDIRIM

Bu çalışmada günümüz şartlarında ulaşılabilen 3 boyutlu örgü sıkıştırma algoritmaları, kütüphaneleri vs. incelenmiş olup akademide bir zamanlar popüler fakat güncel bir koduna, çalışan uygulamasına ulaşılamayan algoritmalar toparlanmış, derlenmiş ve statik üçgensel 3 boyutlu örgü sıkıştırmasında kullanılarak sıkıştırma performansları karşılaştırılmıştır. Ayrıca örgü sıkıştırma algoritmalarının son basamağı olan genel amaçlı veri sıkıştırma algoritmalarının testlere göre günümüz en iyi sıkıştırma istatistiklerine sahip olanları bu derlenen algoritmalar ile beraber kullanılarak toplamda en iyi sıkıştırma oranı elde edilmeye çalışılmıştır. Tek çözünürlüklü 3B örgü sıkıştırma algoritmalarının performansları modern genel amaçlı veri sıkıştırma yöntemleri ile birlikte kullanılarak sonuçlar birbirleri ile karşılaştırılmıştır.

Anahtar Kelimeler: 3B Örgü, Örgü Sıkıştırma, 3B Model Sıkıştırma, Veri Sıkıştırma

GENİŞLETİLMİŞ ÖZET

Geçtiğimiz 20 yılda dijital 3-boyutlu modeller gün geçtikçe daha fazla öneme sahip olmaya başlamıştı. Önemi artan dijital 3-boyutlu modellerin ayrıntısı, doğal olarak boyutları da artmaya başlamıştı. Boyutu artan 3-boyutlu modellerin işlenmesi ve saklanması maliyeti artmıştı. Bu artan maliyet 3-boyutlu modelleri sıkıştırma ihtiyacı doğurdu. İşlemci gücünün çok olmadığı, bellek sıkıntısı yaşanan zamanlarda 3 boyutlu model sıkıştırma araştırmaları başlamıştır. Araştırmalar derinlemesine yapılmaya, sıkıştırma işlemlerinde en küçük kazançlar dahi hesaba katılmaya çalışılmıştır.

Yaklaşık on yıl süren 3 boyutlu modellerin sıkıştırılması konusu, teorik noktaya ulaştığını iddia eden bir araştırmadan sonra daha fazla araştırmacı tarafından yeterince ilgi toplayamamıştır. Gelişen teknoloji ve sınırların esnemesi ya da kalkmasından dolayı sıkıştırmaya olan ihtiyaç önemini yitirmiştir. 3 boyutlu model sıkıştırma konusundaki araştırmalar kademeli sıkıştırma konusuna doğru kaymıştır. Kademeli sıkıştırma yeni çağın gereği haline gelmiştir. Aynı zamanda kademeli sıkıştırma yöntemleri 3 boyutlu modelleri internet üzerinden gönderilebilmeye uygun hale getirmiştir. En önemli özelliği olan 3 boyutlu modelleri kademeli bir biçimde gösterebilmesi kademeli sıkıştırmayı araştırmacıların geliştirmelerini yönelttiği alan haline getirmiştir. Kademeli sıkıştırma algoritmaları tekil 3 boyut sıkıştırma algoritmalarına göre toplamda daha iyi sıkıştırma algoritmaları değildir. Fakat internet çağında bant genişliği gibi yeni kısıtlarla karşılaşıldığında bu kısıtları aşabilecek çözümler ortaya koymuştur.

Araştırma konularının 3 boyutlu modelleri sıkıştırma konusunda kademeli ya da 3 boyutlu model dizileri gibi yöntemlere kayması veri sıkıştırma alanındaki önemli gelişmelerin tekil 3 boyutlu model sıkıştırma algoritmalarına değil de daha çok yeni konuların üzerinde uygulanmasını sağlamıştır.

Bazı sıkıştırma algoritmaları sıkıştırmadan önce veriyi genel amaçlı sıkıştırma yöntemleri ile etkili bir biçimde sıkıştırılabilecek hale getirmeye çalışır.

Bu yüzden 3 boyutlu model sıkıştırma algoritmalarında genellikle son basamak olarak dönüştürülmüş ya da özel bir şekilde sıkıştırılmış 3 boyutlu model verisi genel amaçlı veri sıkıştırma yöntemleri ile tekrar sıkıştırılıp entropi olabildiğince yükseltilmeye çalışılır.

Bu çalışmada literatürdeki tekil 3 boyutlu örgü sıkıştırma algoritmaları, endüstri tarafından geliştirilip kullanılan kütüphaneler, açık ya da kapalı kaynak kodlu yazılımlar, geliştirme araçları, zamanında popüler fakat güncel bir koduna ya da çalışan uygulamasına ulaşılamayan algoritmalar toparlanmaya, derlenmeye çalışılmıştır. Bu uygulamalar, sonrasında tekil 3 boyutlu örgü sıkıştırmasında kullanılarak sıkıştırma performansları karşılaştırılmıştır.

Tekil 3 boyutlu örgü sıkıştırma algoritmalarının son basamağı olarak kullanılabilen genel amaçlı veri sıkıştırma algoritmalarının karşılaştırmalı değerlendirme deneylerine göre günümüz en iyi sıkıştırma istatistiklerine sahip olanları öncesinde belirttiğimiz derlenen algoritmalar ile beraber kullanılarak toplamda en iyi sıkıştırma elde edilmeye çalışılmış ayrıca genel amaçlı sıkıştırma yöntemlerinden hangilerinin tekil 3 boyutlu örgü sıkıştırmaya en uygun yapıya sahip olduğu tespit edilmeye çalışılmıştır.

ACKNOWLEDGMENT

I would like to express my endless thanks and appreciation to my supervisor, Assoc. Prof. Dr. Mustafa ORAL, for all respectable knowledge, support, and patience.

My greatest thanks are for my wife, Gülnur ELMAS, for her support and love.

CONTENTS **PAGES**

LIST OF FIGURES **PAGES**

LIST OF TABLES **PAGES**

1. INTRODUCTION

The contribution of computer graphics to science and technology is always groundbreaking. At the same time, computer graphics are one of the most exciting areas of computer science that attracts researchers. Due to the high number of researches, this field is growing rapidly, possibly more than any other aspect of computer science.

Computer graphics field is incorporation with various domains from scientific applications, engineering, visualization, medical imaging to entertainment, media, game industry etc. Visualization methods with the development of 3D modeling technology have leaped forward. Apart from all these, there is a concept of 3D models that can be considered relatively new in the field of computer graphics. With the help of computer graphics and technological developments in 3D scanners, 3D models began to emerge and computerized visualization and industrial design were opened to another dimension.

Industry began to use 3D models very quickly and enthusiastically. If a picture is worth a thousand words an interactive 3D model is worth a million then. Industrial design, which has a steady place in production stages, has become heavily dependent on 3D models and modeling programs. Since 3D modeling facilitated the process during the production phase, it did not have any problems in providing funds from companies for the development of the technology. Even so, quick answers were produced by the academy and open source community to the needs and problems.

Computer animation is also an important part of computer graphics. It has attracted attention worldwide and has become one of the most successful applications of digital media technology. It has revolutionized computer animation, film world, TV and computer games industry. Many other fields such as marketing, arts, and sciences etc. has been cooperating with computer animation which has shown us the possibilities of creating more realistic and natural scenes, comprehensive simulations of complex problems, and access to places that are difficult or impossible to discover.

3D animation simply gives life to static 3D objects, creating a sequence of static meshes each of which represents one frame. Today, animation technology has also become more sophisticated and accessible. Furthermore, its applications have become more and more well-known and mostly require animated 3D models and scenes with a high degree of realism. It is therefore inevitable to compress 3D datasets.

In this part of the problem, compression algorithms come into play. Limits have always been a repressive element in developing new compression methods, codecs. Within the limits, various codecs have been developed according to the needs. There are a bunch of codecs for image compression which later also adapted for video. So far, compression methods are considered successful in fulfilling the needs.

1.1. Problem Statement

As 3D models, animation or datasets becomes more realistic and more complex, the corresponding 3D meshes' demands getting bigger and bigger, consuming more space which resulting more storage cost, consuming more CPU instruction cycle even causing more cache misses, and most importantly demand more bandwidth when using on the internet.

At first scanning technology can process low-resolution models which are relatively small compared with nowadays 3D models' size. The development of 3D scanning technologies did not take too long to respond. The resolutions of the 3D models increased in detail and resolution, increasing the digital 3D model size as a result.

Lots of exciting ideas and new theoretical approaches have been found a way of reducing the amount of storage for mesh models. Mpeg-4 and Java3D are some of those ideas that become industry standard. There are different needs and different solutions to these needs. The different requirements have led to different solutions that vary between the effectiveness of representation and the accuracy of the details. Approaches can be lossy or lossless, as well as can be progressive and single-rate. Lossy storage is not preferred in some CAD systems, therefore lossless compression methods are retaining their role in 3D compression world.

1.2. Thesis Contribution

In this thesis, currently available 3D mesh compression algorithms, frameworks, libraries etc. are investigated in order to display the advancements and the current state of 3D compression algorithms. Especially, the algorithms that are popular in survey papers but don't have any implementation or had outdated implementation or no published version is available, are gathered together and compiled accordingly. Some algorithms which are mentioned in survey papers (Taubin and Rossignac 1999, Alliez and Gotsman 2005, Peng et al. 2005, Maglo et al. 2015), are not available to end-user or not even published at all. Reaching authors for every method that is not publicly available could not be the case. Implementing from their paper may resolve the problem but while coding original intentions might not be maintained. Different implementations may reveal different programs which may not be reliable for using comparison purposes. Some popular algorithms don't have a compiled version or outdated development environment requirement. This thesis contributed to the field by compiling these algorithms from original source with the updated requirements of new libraries and dependencies which result in working binaries of pioneering 3D mesh compression algorithms. The updated algorithms will be made publicly available to future researchers in the field.

The thesis will also contribute to the field by supplying a comprehensive text material that takes the future researchers to a voyage on the advancements on the field as well as basic understanding of the topic.

According to the benchmark results, current best general-purpose data compression methods are identified and applied as the last stage of mesh compression. Results are compared against each other in order to demonstrate the current state of single-rate 3D mesh compression performance with the current best general-purpose data compression methods.

1.3. Thesis Layout

The layout of the thesis is organized as follows. Chapter 2 provides the basics of 3D mesh concept to familiarize the reader with the terminology. Chapter 3 introduces current data structures for 3D meshes. Chapter 4 making an introduction to the bounds of this thesis before reviewing compression methods in Chapter 5.

Experimental design of our approach and collected mesh compression methods covered in chapter 6. Chapter 7 is the summary of chapter 5 and the results of the work done throughout the thesis mentioned in chapter 6. Chapter 8 is the conclusion chapter which also includes future work can be done to improve or extend this thesis.

2. BACKGROUND AND BASIC CONCEPTS

This chapter introduces the concept of a mesh. In order to understand the 3D mesh and mesh compression methods, definitions and explanations are presented in this chapter.

Among numerous representation methods, an effective way to represent 3D meshes is triangle meshes. Mesh representation can be either in 2D or 3D. However, the real geometry that compression algorithms are dealing with will always be the 2D projection of a 3D model.

2.1. Triangular Mesh

The most basic and simplified representation of the surface is triangular mesh. Mesh representation is heavily handled by triangular meshes in computer graphics related areas like computer-aided design and manufacturing (CAD, CAM). Even polygons can be tessellated to form triangle meshes. Triangular meshes consist of three basic entities: vertices, edges, and faces Figure 2.1. Vertices are points in the 3D world. Vertices are connected by lines called Edges. Edges are formed a closed area called Faces.

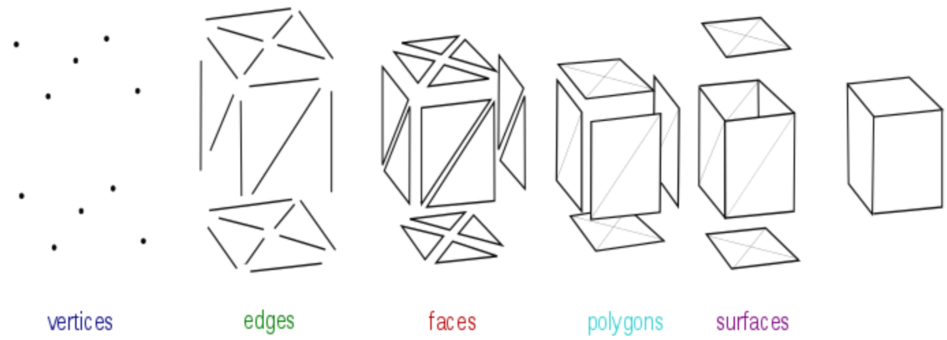


Figure 2.1 Polygonal 3D mesh elements respectively; vertices, edges, faces, polygons, surfaces (Rchoetzlein 2009)

2.2. Manifold Mesh

The edge has to be connected to only two faces in order to be manifold. If an edge is connected to only one face, it's called boundary edge. A mesh is manifold if every edge in the mesh is either a boundary edge or a manifold edge. At the same time, the faces incident to a vertex must form an open or a closed fan.

Another important topological characteristic for a mesh is 2-manifold. If the surface of a mesh is homeomorphic to a disk or a half-disk that mesh is 2-manifold. Non-manifold vertices or edges disrupts the 2-manifold property of a triangle mesh.

A non-manifold edge has more than two faces connected to itself. A non-manifold vertex is the only connection between surfaces or fan of triangles Figure 2.2. Non-manifold meshes are tricky. Most of the algorithms couldn't handle non-manifold meshes cause there is no consistent connectivity information.

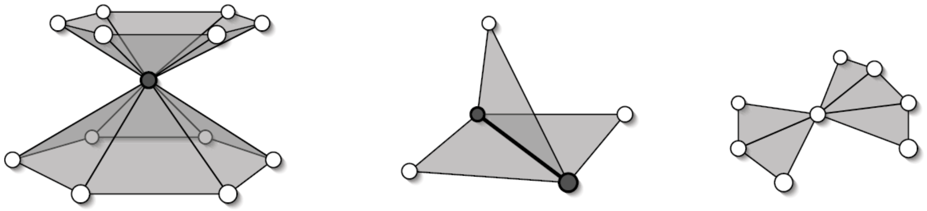


Figure 2.2 Non-manifold vertex (left). A non-manifold edge (middle), a configuration of a non-manifold but can be handled by most of the data structures (right). (Botsch et al. 2006)

A and B can be homeomorphic if A can be extended or bent to B without tearing B. Generally speaking the homeomorphism is a stretching and bending of the object into a new one. A square and a circle, a cube and a sphere are homeomorphic to each other Figure 2.3, but a sphere and a torus are not homeomorphic to each other.

There are interior and exterior edges. Interior edges are not at the border, therefore, they are 2-manifold. Exterior or boundary edges are at the border so that they only connected to one edge.

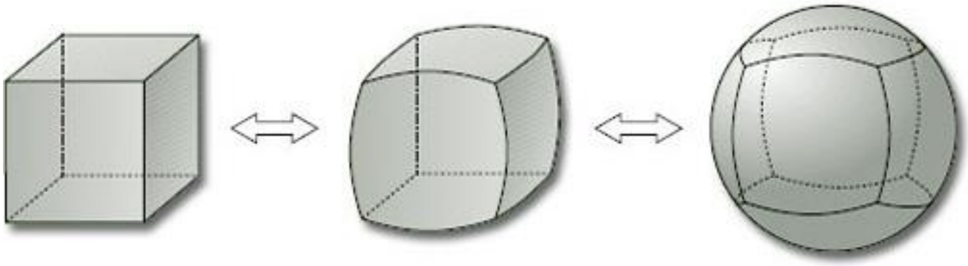


Figure 2.3 Cube and sphere are homeomorphic to each other. ("Homeomorphic surfaces" n.d.)

The simple mesh is a triangle mesh. Simple mesh shapes a manifold, orientable, connected surface. A number of handles define the genus of an orientable connected manifold without boundary. Simple mesh has no handle and either has no boundary.

Regular triangular mesh has a valence of 6 for interior vertices and valence 4 for boundary vertices. Not regular meshes are called irregular or extraordinary. Mesh topology can be irregular, semiregular, or regular Figure 2.5.

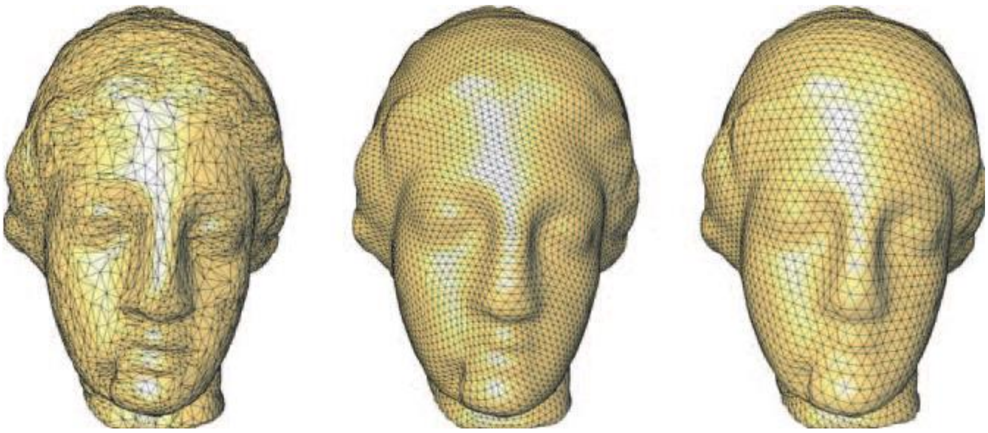


Figure 2.4 Meshes from left to right: Irregular, semiregular, and regular. (Alliez et al. 2008)

2.3. The Euler-Poincaré formula

The Euler formula states a relation between the number of vertices V, edges E, faces F, genus g, connected component c (generally 1), and boundary edges b in a closed connected mesh: **F – E + V = 2(c – g) – b**. Euler characteristic of a model defined as the right-hand side of the Euler formula **χ = 2 - 2g**. Closed 2-manifold polygonal mesh has Euler characteristic of 2 where the genus is 0. Torus which has genus value 1 has Euler characteristic of 0 Figure 2.4.

Since in most real-world applications the genus and border are unimportant compared to the number of elements, the righthand side of Euler formula can be assumed as a trivial. Each face uses 3 edges and each edge is used by 2 faces. Therefore, **2E ≈ 3F**. The number of faces is doubled the number of vertices: **F ≈ 2V**. The number of edges is tripled the number of vertices: **E ≈ 3V**. The average valence for a vertex is 6. Sum of valences is twice the number of edges. These relations come in handy when estimating the runtime complexity of mesh processing algorithms and when analyzing file formats or data structures for triangle meshes.

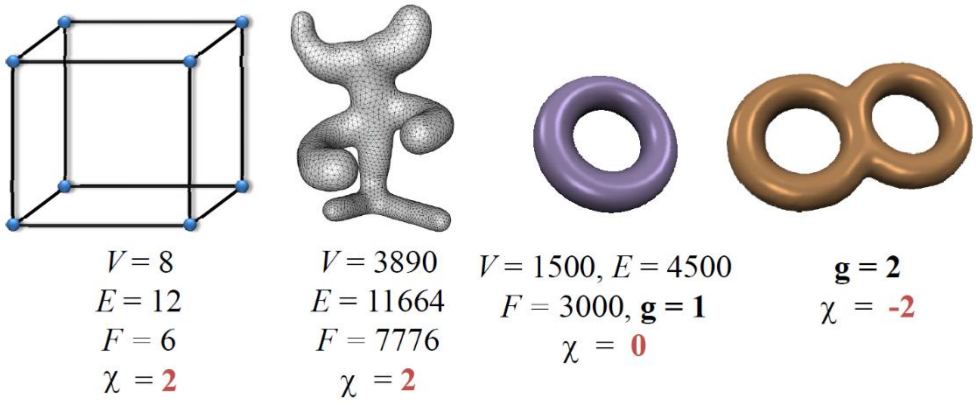


Figure 2.5 Euler characteristic F – E + V = χ

2.4. Connectivity and Geometry

Mesh elements are vertices, edges, and faces. On its most basic form, meshes are represented by geometry and their connectivity (also called topology or structure) information.

Relations of mesh elements are defined by pairs of the same type. Connectivity information contains this relation (adjacency) information.

Geometry describes point locations on 3D Cartesian space for each vertex and may also describe normal vector values for each face.

2.5. Orientation

The orientation of a face is clockwise or counterclockwise order of its vertices. The orientations of two adjacent triangles are called compatible if they execute opposite directions on their common edges. A mesh is orientable if all orientations of its faces are compatible.

For every edge, orientations of two faces that are connected with a common edge have to be a different orientation in order that mesh to be orientable.

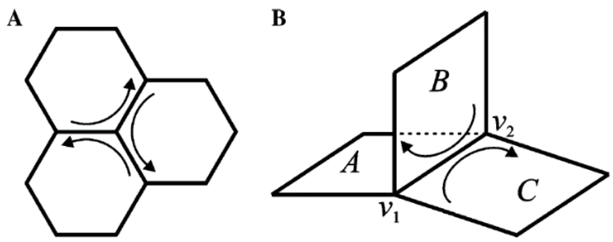


Figure 2.6 Orientable manifold (A). Non-orientable non-manifold (B). Orientable non-manifold (C) meshes. (Peng et al. 2005)

If there is a case for orientations that pairs of triangles are compatible for all, then that mesh is called manifold mesh.

Orientation implicitly holds the inside-outside information of a mesh. This information used for calculating normals. For some data structures traversing the mesh is only possible by orientation information.

Non-orientable mesh has normal calculation problems which result in no inside or outside reliable information. Therefore, it makes difficult to navigate in the mesh.

2.6. Compression Performance

The general convention about compression performance is bit per vertex (bpv). On the other hand, not all papers fulfill this convention and use bit per triangle (bpt) and even use the total compression ratio (i.e. 1:17). In this thesis, popular algorithms' bpv information has been given. While comparing with the current best algorithms, libraries, frameworks etc. bit per vertex convention have been used as far as possible. However, in this thesis we have provided four output types for comparison purposes: bit per vertex, space saving that method can provide in percentage, storage cost after compression according to raw data in percentage and finally the compression ratio.

3. DATA STRUCTURES OF MESH

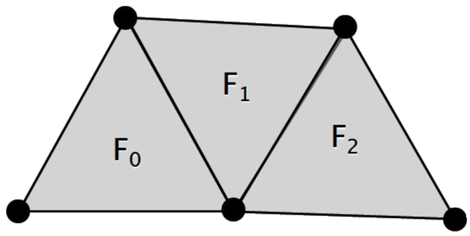
Mesh representation and implementation problems yield to different kind of data structures that are listed below. Each of these data structures is a solution to domain-specific problems like implementation easiness, compactness, efficient traversal, etc. Most importantly data structures should offer an efficient way of retrieving different kinds of adjacency information fast and without taking up much storage space.

Data structure should consider some design criteria to overcome above domain specific problems. One of the main criteria is the storage cost. Data to be stored must be carefully selected for storage efficiency. Some information can be generated from each other. However, geometry information consists of 3D coordinates of vertices and attribute information consist of normal, color, texture coordinate of every vertex, face, edge need to be stored explicitly. On the other hand, connectivity information can be stored in a various way that's why it is the most studied data by most of the data structures. Other criteria data structure should support are rendering, geometry queries, modifications, and compression availability.

Data structures should support basic operations resourcefully. For a given face it should find its vertices and neighboring faces. For a given vertex it should find face touching it and neighboring vertices etc.

3.1. Face Set

Also known as Independent Faces or Separate Triangles, Face Sets just store a list of faces Figure 3.1. For each face, store positions of its vertices are stored. There is no connectivity information but a collection of polygons. 3D print file extension STL uses Face Set data structure. Face set structure does not need to store all data in memory to render the mesh. However, redundancy is excessive.

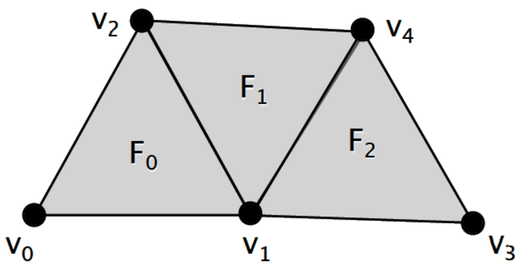


Face Table
F_0: (x_0,y_0,z_0), (x_1,y_1,z_1), (x_2,y_2,z_2)
F_1: (x_3,y_3,z_3), (x_4,y_4,z_4), (x_5,y_5,z_5)
F_2: (x_6,y_6,z_6), (x_7,y_7,z_7), (x_8,y_8,z_8)

Figure 3.1 Face Set - Independent Faces - Separate Triangles

3.2. Indexed Face Set

Indexed face set also known as an indexed structure or shared vertex. A triangular mesh can be represented with shared vertices that consist of a vertex coordinate array and a face array. The face array registers face by indexing its vertices in the coordinate array which registers the coordinate of all vertices Figure 3.2. Connectivity information encodes through face array. In this data structure, each vertex is shared multiple times by all its incident triangles. The whole list of vertices needs to be stored in the memory.



Vertex Table
v_0: (x_0,y_0,z_0)
v_1: (x_1,y_1,z_1)
v_2: (x_2,y_2,z_2)
v_3: (x_3,y_3,z_3)
v_4: (x_4,y_4,z_4)

Face Table
F_0: 0, 1, 2
F_1: 1, 4, 2
F_2: 1, 3, 4

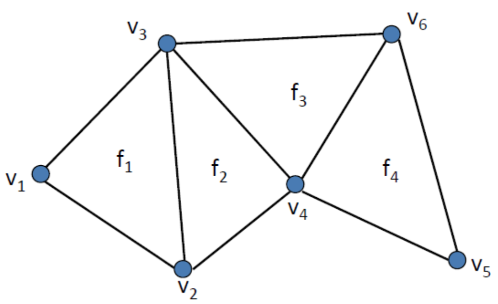
Figure 3.2 Index Face Set – Indexed Structure – Shared Vertex

Indexed Face Set data structure is easy to implement and quite compact but not efficient for traversal. Raw file formats (OBJ, PLY, OFF, WRL, etc.) are inspired from indexed data structure while representing meshes.

3.3. Adjacency Matrix

This data structure model stores the mesh connectivity information in the adjacency matrix. If there is an edge between $\mathbf{v_i}$ and $\mathbf{v_j}$ then $\mathbf{A_{ij}}$ = 1 Figure 3.3. New features can be accessed thanks to adjacency matrix representation. For example: $\mathbf{(A^n)_{ij}}$ = Number of paths whose length is n from v_i to v_j.

Adjacency Matrix can represent non-manifold meshes. On the other hand, store no connection between a vertex and its adjacent faces.

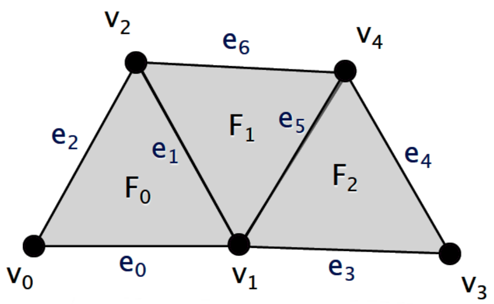


A =

	v_1	v_2	v_3	v_4	v_5	v_6
v_1		1	1			
v_2	1		1	1		
v_3	1	1		1		1
v_4		1	1		1	1
v_5				1		1
v_6			1	1	1	

Figure 3.3 Adjacency Matrix data structure representation

3.4. Adjacency Lists



<u>Edge Adjacency Table</u>

e_0: v_0, v_1; F_0,∅; ∅,e_2,e_1,∅

e_1: v_1,v_2; F_0,F_1; e_5,e_0,e_2,e_6

⋮

<u>Face Adjacency Table</u>

F_0: v_0,v_1,v_2; F_1,∅,∅; e_0,e_2,e_0

F_1: v_1,v_4,v_2; ∅,F_0,F_2; e_6,e_1,e_5

F_2: v_1,v_3,v_4; ∅,F_1,∅; e_4,e_5,e_3

<u>Vertex Adjacency Table</u>

v_0: v_1,v_2; F_0; e_0,e_2

v_1: v_3,v_4,v_2,v_0; F_2,F_1,F_0; e_3,e_5,e_1,e_0

⋮

Figure 3.4 Full Adjacency List data structure representation

Information is stored in Adjacency Lists which can be full of adjacency of only required part depending on the developer.

Full adjacency list stores all vertex, edge, and face adjacency Figure 3.4 without considering redundancy or storage cost. Traversal is easy. Querying mesh is effortless. However, modification of a mesh requires updating lots of data.

Partial adjacency list stores only part of the full adjacency list and derived others from redundancy Figure 3.5. According to the data chosen to be stored some combination only work on specific meshes.

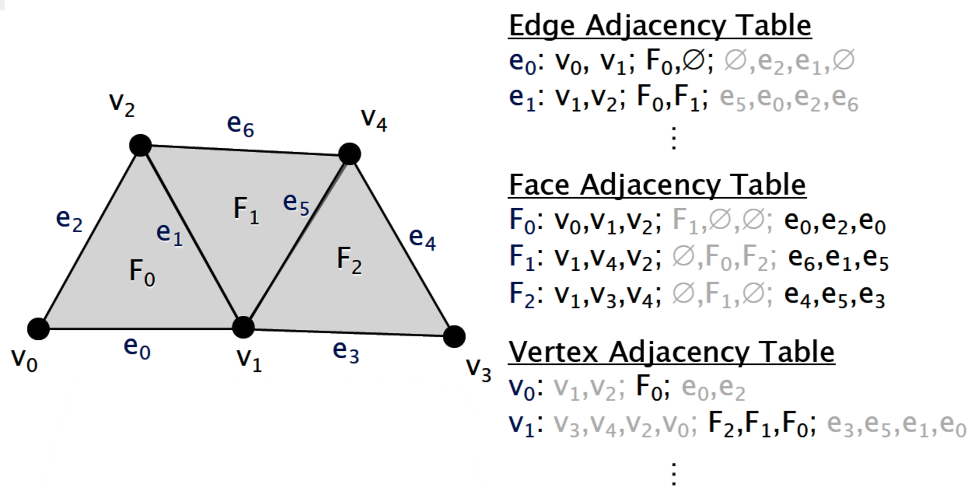


Figure 3.5 Partial Adjacency List data structure representation

3.5. Winged-Edge

Winged-Edge representation can also be named under the partial adjacency list but winged-edge stores most of the information on edges and derives face and vertex adjacency from edge adjacency. What is crucial about winged-edge representation is every face and vertex adjacency table can only point to one edge. Each edge is fixed size: 2 vertices, 2 faces, and 4 edges. Winged-Edge representation has enough information to traverse. All topological relation can be retrieved in optimal time. Being edge-centric rather than face-centric generalizes it to work with polygonal meshes.

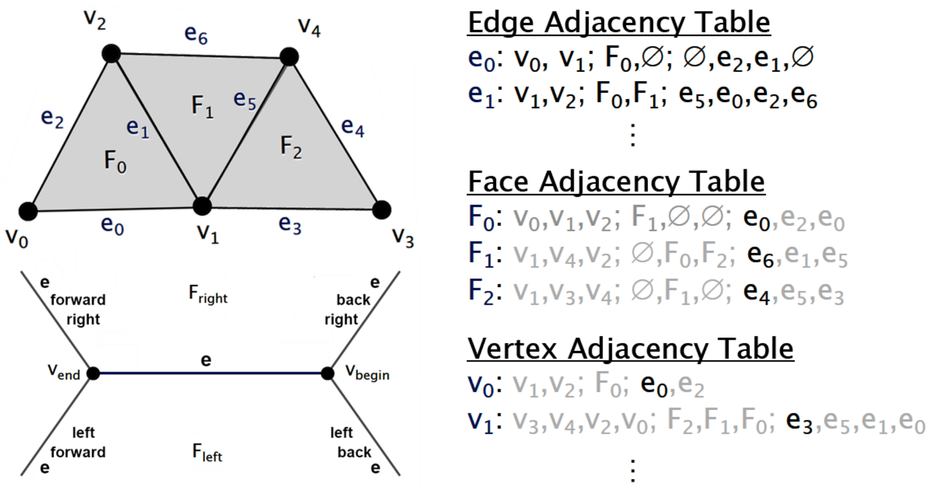


Figure 3.6 Winged-Edge data structure representation

3.6. Half-Edge

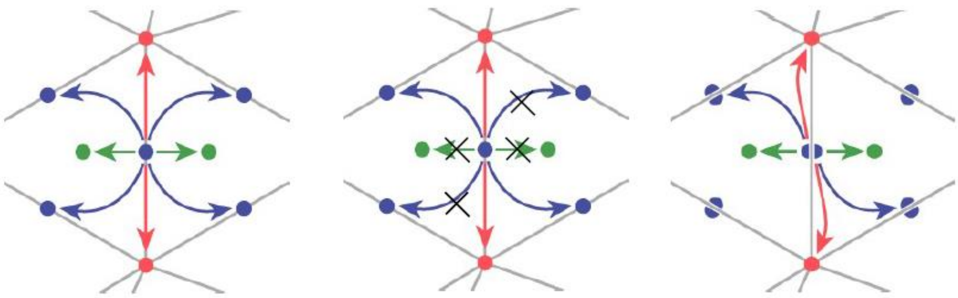


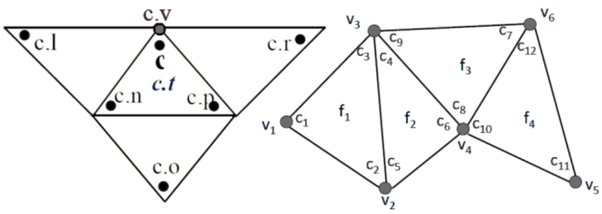
Figure 3.7 Respectively: Winged-Edge, Optimized Winged-Edge, and Half-Edge

Winged-Edge representation optimized to Half-Edge representation by using 2 half-edges instead of a single edge. So that an edge corresponds to a pair of half-edges with opposite orientations Figure 3.7. Each half-edge stores half topological information concerning the edge. Optimization applied to Winged-Edge data structure by omitting faces if not needed and also by omitting one edge pointer on each side which results in one-way traversal called Half-Edge data structure.

Half-Edge representation is edge-centric which enables generalizing it to work with polygonal meshes. Efficient traversal and update operations provided.

3.7. Corner Table

Corner table is yet another simple data structure which makes it easy to process and store of manifold triangular meshes. Corner table data structure consists of G table, V table, and O table Figure 3.8. Coordinates of vertex v stored in the G[v] table, represented as **v.g** in (Rossignac 2005).



corner	c.v	c.t	c.n	c.p	c.o	c.r	c.l
c_1	v_1	f_1	c_2	c_3	c_6		
c_2	v_2	f_1	c_3	c_1			c_6
c_3	v_3	f_1	c_1	c_2		c_6	
c_4	v_3	f_2	c_5	c_6		c_7	c_1
c_5	v_2	f_2	c_6	c_4	c_7	c_1	
c_6	v_4	f_2	c_4	c_5	c_1		c_7

Figure 3.8 Traversing the mesh with a corner table operators and example

Vertex-Triangle relation defines each triangle by integer references of three which are its vertices. These references are kept as successive entries in the V table. Although G and V are enough to identify the triangles and the triangle they represent, accessing to a neighbor triangle or vertex can't be established with the only G and V. Problem resolved with O table, opposite corner representation. Accessing other elements with corner table representation is straightforward like: Corner (**c**), Triangle (**c.t**), Vertex (**c.v**), Next corner in c.t (**c.n**), Previous corner (**c.p**), Corner opposite (**c.o**), right corner (**c.r**) and left corner (**c.l**).

All geometric queries result in O(1) time. Most operations are O(1) which make corner table convenient for rendering operations and hardware implementations. On the other hand, corner table only works with manifold triangular meshes. Since corner table has high redundancy, many derived versions developed by (Gurung and Rossignac 2010, Gurung et al. 2011a, 2011b, 2013, Luffel et al. 2014).

Sorted Opposite Table (SOT) matches each vertex with a different triangle and rearranges triangles so that triangle x of the first m triangles matches to vertex x, where m is the number 12 of vertices. So, there is no need to store the incidence V table, which can be recovered by swinging around each vertex till a triangle is reached with a sufficiently small identifier. The **v.c** operator is also available implicitly.

Sorted Quad (SQuad) extends SOT by pairing most unmatched triangles with matched vertex-triangle combination to form matched quads. By forming quads, SQuad avoids storing one opposite corner per triangle between triangles in the same quad.

LR rearranges the vertices and matched incident quads of a mesh along a ring which is a nearly Hamiltonian cycle. It links the two triangles incident on a (directed) ring edge e with the vertex v at the source of the edge. LR then stores, for each v, the (integer) references **v.L** and **v.R** to the tip vertices (those not on **e**) of the two triangles that these vertices form with e. Given that most of the corners are in the ring, this means storing a reference (32 bits) per triangle. In LR, many adjacency relationships can be inferred from the ring.

Zipper extends LR and prevents storing most of the **v.L** and **v.R** references directly. Instead, it stores a pair of 3-bit codes for most ring vertices. These codes store delta increments between two **v.L** (**v.R**) consecutive entries from which **v.L** (**v.R**) are derived in constant time.

Similarly, Grouper extends SQuad representation, Grouper shows the geometry and connectivity of a mesh by storing two adjacent triangles and a common vertex, grouping the corners and triangles in fixed size records. Unlike SQuad, Grouper inserts the geometry data within connectivity data and uses a new connectivity representation to show that corners and triangles can be stored in a consistent order.

3.8. Summary

Implementing data structures for 3D mesh may look like an easy programming job at first look, actually, it is much harder to balance between flexible, memory-aware, and computation efficient data structure.

Different data structures have been overviewed by the (Kettner 1999) and by the (De Floriani and Hui 2003). There are other data structures, for further reading specialized for a range of tasks and size of data. (Isenburg and Lindstrom 2005) has processing massive meshes and (Cignoni et al. 2004) has a view-dependent rendering of massive meshes. Data structures need to decide between low memory usage or full access. For this decision kindly refer to the (Kallmann and Thalmann 2001, Castelli Aleardi et al. 2008).

4. CLASSIFICATION

3D mesh compression is always an essential field for the future of multimedia compression. Remeshing, simplification, and compression: these are the three main approaches to reduce the size of a 3D mesh. The purpose of the compression approach is to have a coding bit stream as short as possible so that the compressed file size becomes as small as possible.

Compression is also valuable as an encoding tool for simplification and remeshing approaches that result in a small mesh, which is often required to efficiently encode large databases with many models. Particularly regular and large models usually contain more information than required or redundant information. While preserving the connectivity of the mesh, simplification should be performed.

The widely accepted idea in network simplification is that the network is simplified by a number of local processes that remove a small number of adjacent mesh elements. Remeshing is also a promising approach in compression area. A regular mesh is approximated to original mesh. Exploiting this regularity of approximation ensures efficient storage of a mesh.

3D models are generally polygonal meshes, in this thesis, the focus will be mainly on compression techniques for 3D triangular meshes which is also a polygonal mesh.

Typically, connectivity, geometry and attribute data are enough to represent and define a 3D mesh. Geometry data specify vertex locations in 3D space. Connectivity data holds the neighborhood information between vertices. Attribute data states other properties such as normal vectors, material information, and texture coordinates etc. Therefore, according to which part of 3D triangular mesh data is going to be compressed, 3D mesh compression methods have been grouped into three categories, geometry compression, connectivity compression, and attributes compression.

Most of the well-known compression algorithms' (Taubin and Rossignac 1998, Touma and Gotsman 1998, Rossignac 1999, Alliez and Desbrun 2001a) encoding phase are done separately. There also exist geometry driven connectivity-oriented algorithms too(Lee et al. 2002). Early works focused on the connectivity coding. However, geometry data takes up more bits than connectivity data, and researchers are well aware of the situation and working on compression of geometry data efficiently.

Geometry compression is classified into two classes; lossless and lossy geometry compression whether the reconstructed(decoded) data exactly the same as the original or not. Completely restoring the original geometry data from the compressed data can only be done by lossless compression methods. On the other hand, lossy compression couldn't restore original data. Generally, information is lost in the quantization phase.

3D mesh compression can be performed on spatial-domain or transform-domain. Some networks require data compression to reduce the latency and then select progressive representation to convert a 3D mesh into streams that can be easily managed by networks.

Decoding phase determines the classification of mesh compression methods. If the decoding is started after the transmission, it classified as single-rate (single-resolution mono-resolution) compression. If the decoding is started during the transmission, it classified as progressive compression.

Single-rate 3D mesh compression methods generally create single bitstream which consists of connectivity information that describes the mesh topology and geometry information.

Bitstreams of progressive transmission have several components in it which need to be separated. Typically, both bitstreams contain base mesh which later refined by reading latter bits from the stream.

Lossless coding done by single-rate methods aims to remove the redundancy available in the original data. Progressive compression has to make a tough choice between the data size and accuracy. Accuracy can also be called rate-distortion trade-off.

Lossy coding of single-rate compression can be accomplished by modifying the model, making it easy to handle by codes without degrading too much valuable information.

Early researches on 3D mesh compression mainly concentrated on single-rate methods to reduce the bandwidth usage between the GPU and CPU. Single-rate mesh compression algorithms treat geometry and connectivity data as a whole. In single-rate compression, the rendering process cannot start until the entire compressed data reaches to the decoder.

The popularity of Internet force researcher to work on progressive compression and transmission intensely. With the help of progressive compression, the 3D mesh can be rebuilt continuously from a different level of details while the bitstream is being received. From the development trends perspective, focus on 3D mesh compression techniques are changed from connectivity-driven methods to geometry-driven methods.

Apart from the progressive compression schemes, there is also randomly accessible mesh compression schemes under the static mesh title. Random access algorithms are capable of decompressing only the requested part of the mesh to save resources. This kind of algorithms specifically designed for a model that doesn’t fit in device memory.

Apart from the single-rate compression methods, there are progressive and randomly accessible algorithms available for single-rate compression, but they are generally not as effective as pure single-rate methods.

This thesis will be limited with the specified algorithms of categories and subcategories accordingly; static single rate heavily connectivity-driven, some geometry-driven triangular 3D mesh compression algorithms Figure 4.1.

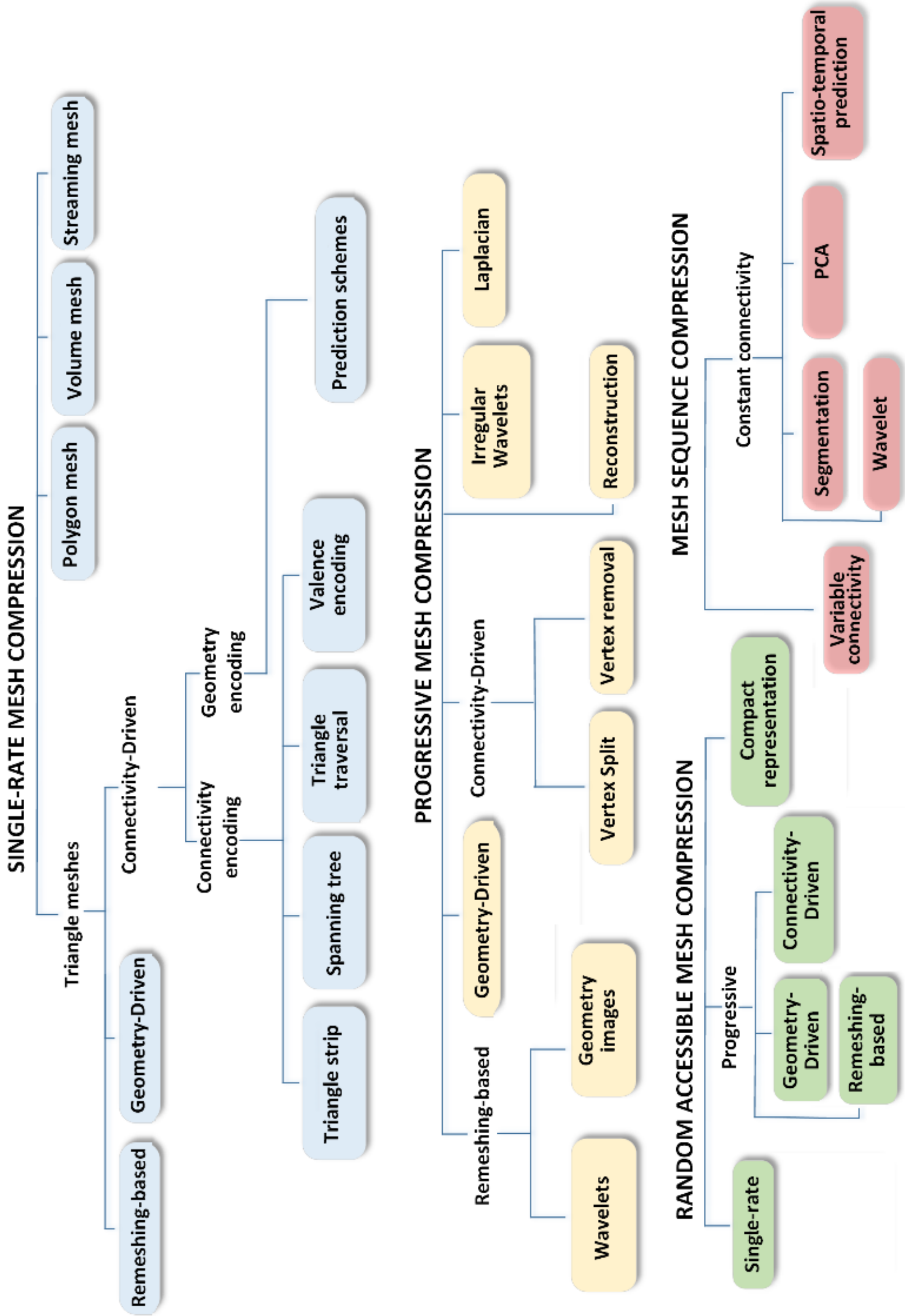


Figure 4.1 Classification of algorithms according to (Maglo et al. 2015)

5. SINGLE RATE MESH COMPRESSION

Multimedia data types, such as audio, image, and video, have common features in which data structures are known by both the encoder and the decoder. However, a mesh structure is generally not defined in advance. The mesh structure is not completely known by the encoder prior to compression. Therefore, in addition to having to encode the geometry, a mesh compression algorithm must also encode the connectivity information.

5.1. Connectivity Compression

5.1.1. Triangle Strip

Triangle strips are easy to handle and well supported by GPUs. The primary purpose of the triangle strip is to reduce the bandwidth usage between central and graphics processing units. The efficiency of the triangle strip method is better than raw formats, like index face sets, but still not very efficient on the compression side. In order to achieve better compression a popular method, generalization applied. Due to its structure, triangular stripes always accept vertices in the same order. However, generalized triangle strips do not always obey this order to create longer strips for exploitation purposes which disrupt its structure.

Generalized triangle strips are a mixture of triangle fans and strips. In an indexed face set, a triangle is recorded by three vertices, in generalized triangle strips triangle is added by only one vertex except for the first triangle.

So, generalized triangle strips provide compressed representation than indexed face sets, particularly if the strips are long. A number of triangles to the vertices ratio is very close to 1 in long enough strips. Which means a triangle can be represented by 1 vertex index accurately by generalized triangle strips.

A mesh typically has twice as many triangles as vertices. Some vertices indices information is repeated in the generalized triangle strip representation, which states a redundancy in storage. To overcome this storage waste number of schemes have been created which use a buffer to hold the indices of recently processed vertices.

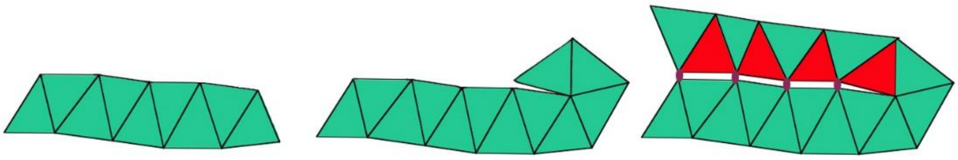


Figure 5.1 The triangle strip (Left), the triangle fan (Middle), and the generalized triangle strip (Right).

(Deering 1995a) first introduced the geometry compression schema to compress the data sent between CPU and GPU by generalizing triangle strips and fans. The new concept used by Deering, generalized triangular mesh Figure 5.1, applied by combining vertex buffer with generalized triangle strips. An example of geometry compression schema of Deering can be seen in Figure 5.2.

The first stage of Deering's method is to convert generalized triangle strip data to generalized triangle mesh while using mesh buffer consisted of 16 slots queue referenced by 4-bit index. Conversion explicitly pushes vertices onto mesh buffer for reuse.

After representing connectivity information with generalized triangle mesh positions, normals and colors quantized to 16-bit as a second stage. According to the Deering 16-bit per component is visually indistinguishable. In many cases, far fewer bits are needed. Geometry is generally local within the 16-bit or less. In the generalized mesh buffer, it is likely that the delta difference between one vertex and the other is significantly less than 16 bits. Like positions, colors also quantized as well but with less accurately to only 12-bit.

Delta encoding applied to neighbors as a third stage to the quantized values. As stated earlier most geometry is local and delta encoding these local values reduce the representation size of quantized values. Depend on delta encoding Deering stated that far fewer bits needed. At the last stage, Huffman based variable-length encoding performed on deltas. All of these stages have specific instructions defined in Deering's original paper as Geometry Compression Instructions. In the end, there is binary stream output with Huffman table initializations and geometry compression instructions.

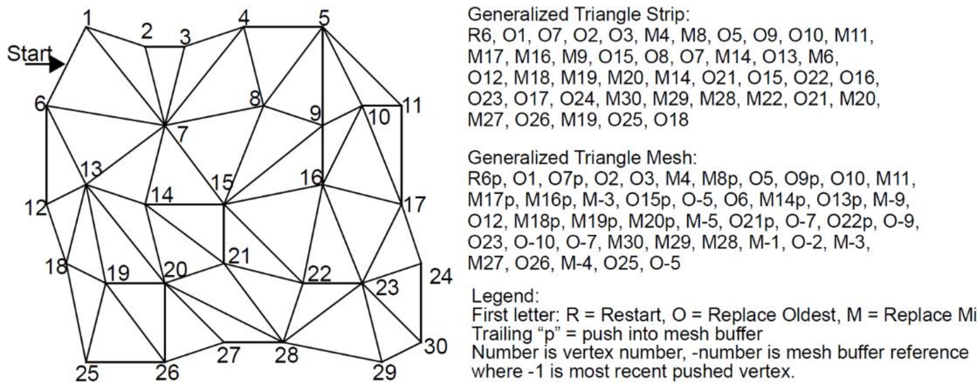


Figure 5.2 Generalized Triangle Mesh (Deering 1995a)

Chow proposed an optimized geometry compression schema based on the original work of Deering, generalized triangle mesh, but specifically optimized for real-time rendering (Chow 1997). Deering's original work didn't show the decomposition of mesh which is later proposed by Chow. Chow's algorithm is influenced by the spiraling traversal of the mesh as (Taubin and Rossignac 1998) in the Topological Surgery compression algorithm. Chow's method applied by decomposing mesh according to Figure 5.3. At first, it finds boundaries. Later, it finds fans around each vertex that is adjacent to two successive boundary edges. These composed triangle fans create the first generalized triangle strip. Then this strip marked as discovered. A new set of boundary edges is selected accordingly separating discovered triangles from undiscovered ones. A new generalized triangle strip is similarly formed from the new set of boundary edges again. Chow also use vertex buffer method, so that the vertices in the previous generalized triangle strip can be reused without wasting storage space. Mesh is traversed until triangles are all processed.

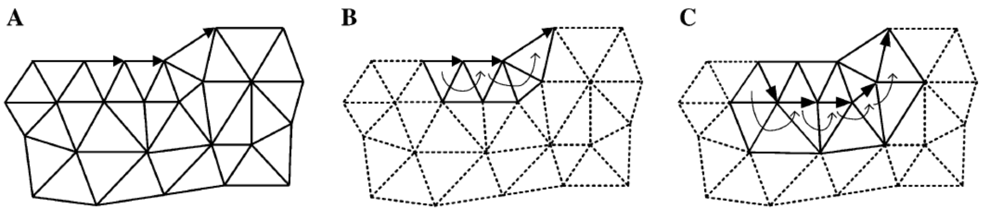


Figure 5.3 Arrows indicating a set of boundary edges (A), triangle fans for the first strip (B), triangle fans for the second strip (C), thick arrows used for selected boundary edges, thin arrows used for the triangle fans associated with each inner boundary vertex. With courtesy of (Peng et al. 2005)

An alternative representation has come from (Bajaj et al. 1999) based on a decomposition of the mesh into triangle and vertex layers used by (Taubin and Rossignac 1998) originally in their Topological Surgery algorithm with a different version. Vertex layers are non-crossing strings of vertices. Usually, they are separated by one edge and take shape of a same-origin circle (concentric) on the mesh. Between the vertex layers, there are the triangle layers which consist of triangle strips and fans. Non-manifold meshes can be handled by this approach.

5.1.2. Spanning Tree

(Taubin and Rossignac 1998) introduces the Topological Surgery algorithm, first introduced as a single-resolution manifold *triangle* mesh compression scheme. Later extended its capabilities to handle arbitrary manifold *polygonal* meshes with properties. VRML standard use Topological Surgery as a binary version of a VRML (VRML Compressed Binary format) (Taubin et al. 1998). Topological Surgery has become a part of the ISO/IEC multimedia standard with more efficient encoding by Moving Picture Experts group, MPEG-4/3DMC. The 3DMC algorithm is a single rate compression method for manifold triangular meshes. Block diagram of a 3DMC encoder can be seen in Figure 5.4.

VRML raw file format represents 3D mesh in ASCII. (Taubin et al. 1998) created a compressed binary format for VRML based on Topological Surgery scheme for efficient transmission. Compressed binary data stream consists of encoded vertex graph, encoded simple polygons, encoded geometry and property data which are quantized, predicted, and compressed. What makes Topological Surgery so efficient in compression lies behind each of these elements encoding process and order.

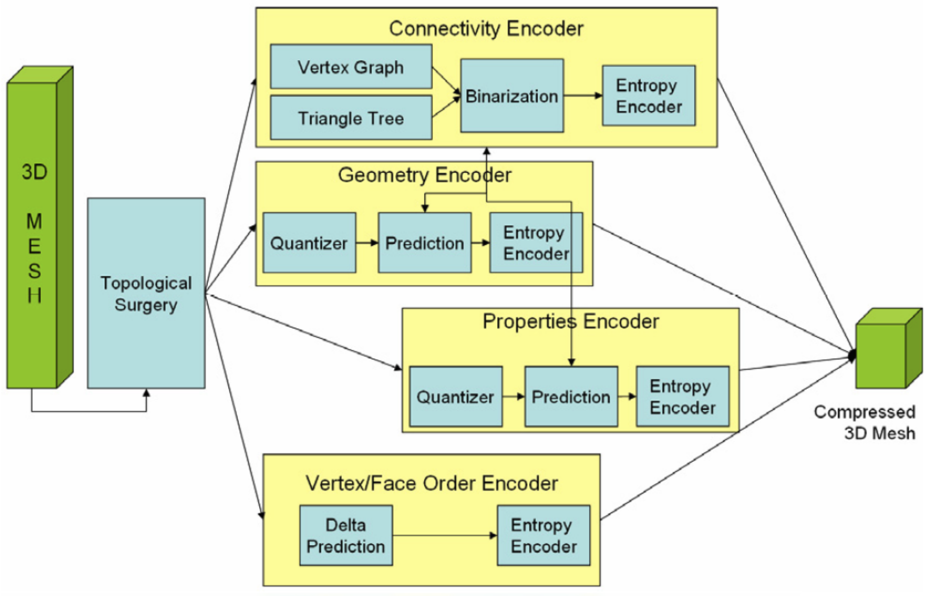


Figure 5.4 Block diagram of a MPEG-4 3DMC encoder (Jovanova et al. 2008)

MPEG-4 implementation of Topological Surgery partitioned connectivity information into per-triangle and global information which is transmitted first. Geometry and property data kept interleaved way in per-triangle information which later transmitted correspondingly.

By using two spanning trees, connectivity of a planar graph able to encoded with constant bit per vertex (Turán 1984). These spanning trees are vertex and triangle spanning tree. To encode mesh connectivity Topological Surgery presented based on this thought which is to make a planar polygon from a selected set of cut edges. The connectivity then represented by the structures of a polygon and cut edges. Set of cut edges need to be selected and any vertex tree in a simple mesh can be selected. The coding cost for both vertex and triangle spanning trees are relational with the number of runs. Vertex spanning tree construction defines the number of runs which is build based on layered decomposition, a spiral path Figure 5.6, to maximize the length of each run so that minimizing the number of runs generated.

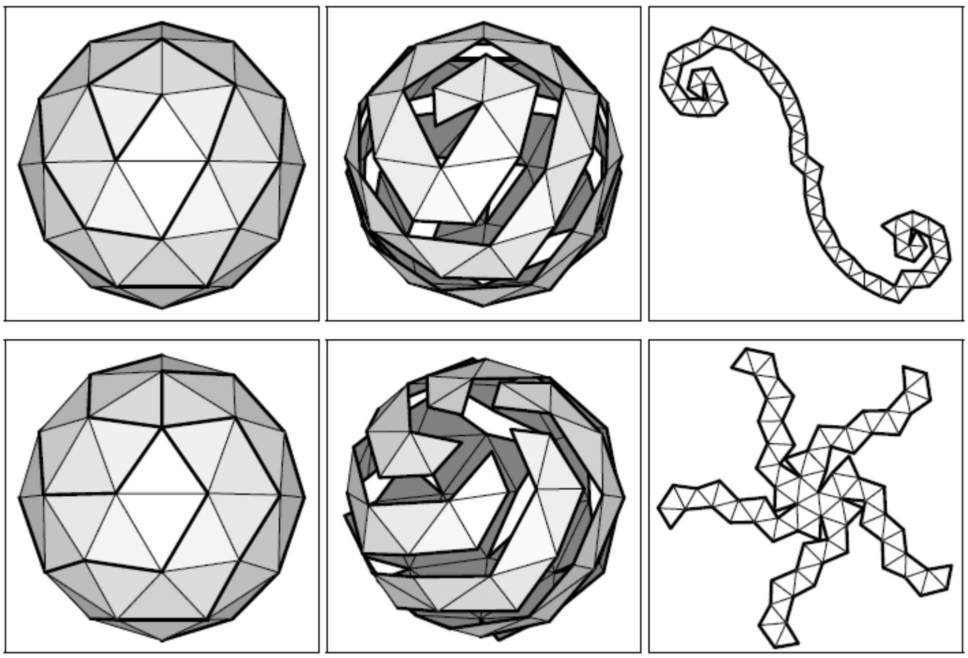


Figure 5.5 Two way for a spiral path (Taubin and Rossignac 1998)

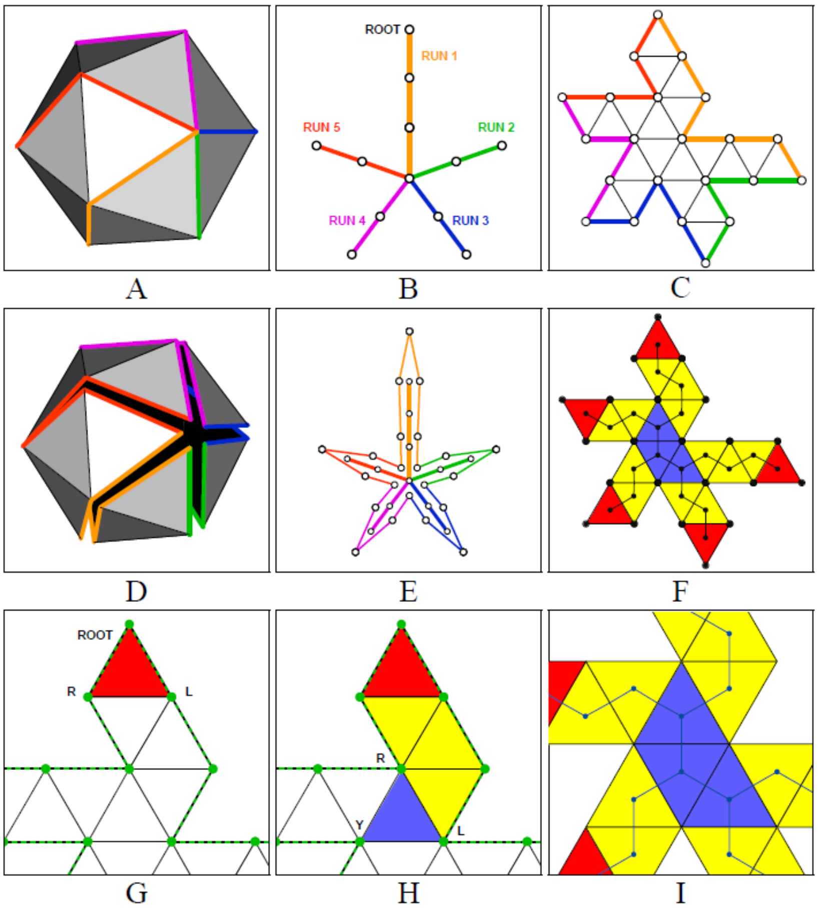


Figure 5.6 Topological Surgery Representation. (Taubin and Rossignac 1998)

Topological Surgery algorithm representation stated in Figure 5.5. The vertex spanning tree (A, B) compiled of vertex runs. Cutting through the vertex tree edges generates topological simply connected polygon (C, D). The bounding loop (E) is the boundary of the polygon. The dual graph of the polygon is the triangle spanning tree (F). Triangle runs end in leaf or branching triangles. Leaf triangles are red, regular triangles are yellow, and branching triangles are blue. The triangle spanning tree has a root triangle (G). Marching edges (H) connect consecutive triangles within a triangle run. Each branching triangle has a corresponding Y-vertex. Two consecutive branching triangles define a run of length one (I).

(Diaz-Gutierrez et al. 2005) presented the Hand-and-Glove algorithm which is a simple variant of a vertex spanning tree based on Topological Surgery algorithm. The Hand-and-Glove algorithm encodes a manifold mesh with the help of two vertex spanning trees: Hand and Glove trees which are built for traversing the entire mesh, in order to form a triangle strip loop. Conceptually Glove trees wrap around the Hand trees in defining the triangle strip. Both spanning trees can be represented by any start node and a depth-first or breadth-first tree traversal. The traversal encoded per node by two bits: one bit for child nodes, and one bit for siblings, if the node has one or more. Every triangle is a member of either the Hand or The Glove tree. The triangle strip is defined by a single start triangle and instructions. Advance to the next following triangle instructions is consist of taking the next vertex either from the Hand or the Glove vertex spanning tree.

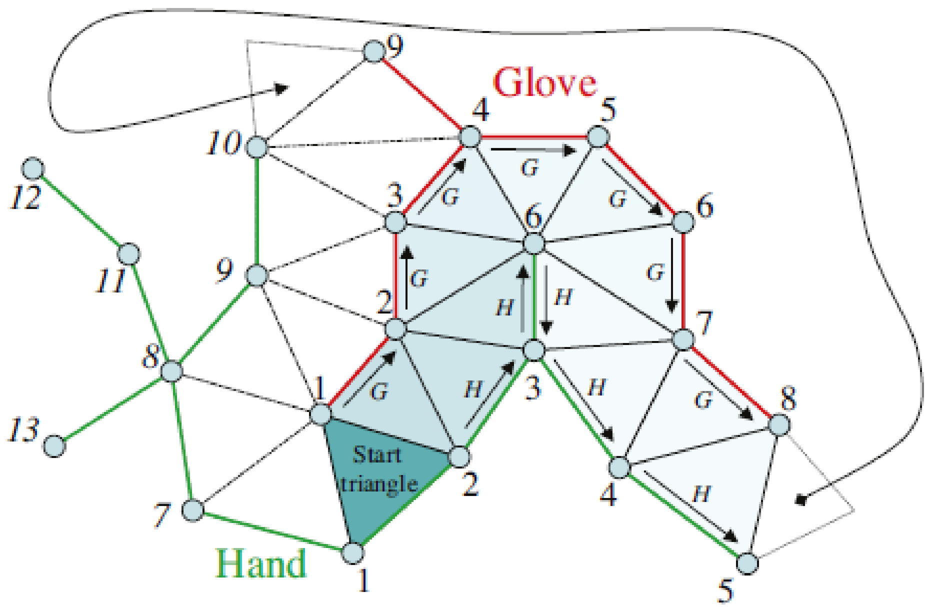


Figure 5.7 Illustration of The 'Hand' and 'Glove' vertex spanning trees traversing the mesh. (Diaz-Gutierrez et al. 2005)

(Li and Kuo 1998) proposed a so-called “dual" approach that traverses the edges of the dual graph (Figure 5.7) and outputs a variable length sequence of symbols based on the type of a visited edge. The final sequence is then coded using a context-based entropy coder.

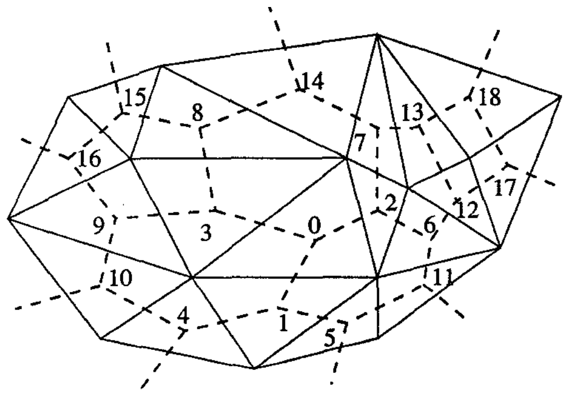


Figure 5.8 Solid lines: A triangular mesh. Dotted lines are its dual graph. (Li and Kuo 1998)

5.1.3. Triangle Traversal (Conquest)

Algorithms that are proposed with a region-growing approach to encoding a mesh by generating a triangle spanning tree, decomposed the meshes into strips. Generation of this spanning trees forces the algorithm iteratively process the mesh triangles with a breadth-first traversal. The simplicity of implementation and ease of understanding are key advantage point for triangle traversal methods. Therefore, triangle traversal methods became the center of researches on mesh compression. There are two main methods: EdgeBreaker (Rossignac 1999) and The Cut-Border Machine (Gumhold and Straßer 1998) each describe a set of growing operations for triangular meshes.

The Cut-Border Machine proposed as one of the first triangle conquest approaches. (Gumhold and Straßer 1998) This approach starts from the initial borderline, which divides the whole mesh into conquered and unconquered parts, and inserts triangles one by one into the conquered parts. Insertion applied by using one of the five (later six) operations: new vertex, connect forward, connect backward, split, close, and later union Figure 5.9. Each operation represented by a symbol: *, →, ←, ∞, ▽, and later ∪. These operations sequence is encoded later using Huffman coding. The Cut-Border Machine can encode manifold meshes either orientable or non-orientable. Its most desirable feature is that the decompression speed is fast and the decompression method is easy to implement in hardware. Compression and decompression can be in parallel as well. These features make the Cut-Border Machine very attractive especially in real-time coding applications.

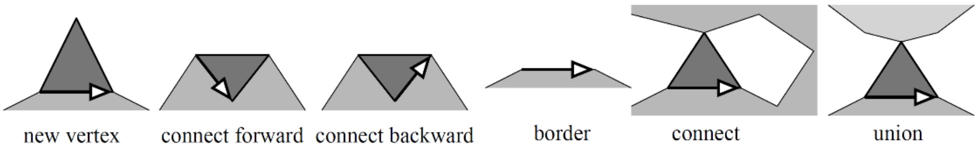


Figure 5.9 Different cut-border operations. The gate is shown as an arrow and the new triangle is shaded darkly.

Triangle conquest has another popular approach called EdgeBreaker proposed by (Rossignac 1999). It is almost equivalent with the Cut-Border Machine. EdgeBreaker algorithm, on the other hand, guarantees the cost of connectivity with its predefined fixed format. Every triangle in the mesh represented by one symbol there is at most five symbols and no other additional offset. Faces are iteratively processed while encoding connectivity on EdgeBreaker.

The triangle conquest is controlled by edge loops. Each loop defines a boundary around the conquered region and contains a gate edge, called the active gate. Initially, there is one edge loop for each connected component. This algorithm focuses on one edge loop et every main iteration. The other edge loops are stored and waiting in the stack to be processed. If the component has no boundary, one edge split into two half-edges and considered as the edge loop.

Table 5.1 Translation between Cut-Border-Machine and EdgeBreaker symbols

Cut-Border-Machine		EdgeBreaker	
Name	Symbol	Name	Symbol
New Vertex	*	Create	**C**
Connect Forward	→	Right	**R**
Connect Backward	←	Left	**L**
Split	∞	Split	**S**
Close	∇	End	**E**
Union	∪	Merge	**M**

An active gate is the start point for conquering a triangle at each step. When a triangle is conquered algorithm updates the current edge loop also update the active gate to the next edge in the loop. For each conquered triangle, this algorithm outputs a symbol. In the original EdgeBreaker algorithm there are at most five kinds of possible symbols for the connectivity structure. In Fig. 5.10 v is the center vertex, and X is the current triangle. The active gate is the lowermost edge in Figure 5.10. The complete fan is configured as symbol C (Create). Configuration symbol of L (Left) stands for active gate has missing triangles at the left. Like in L but another way of it, configuration symbol of R (Right) stands for active gate has missing triangles at the right. When the active gate doesn't have any other triangles, it means configuration symbol is E (End). If there are missing triangles rather than left and right of the active gate, configuration symbol S (Split) is generated.

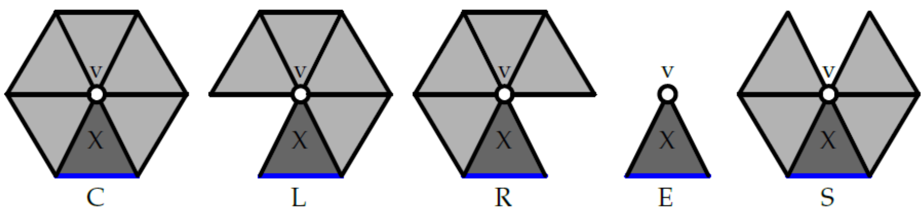


Figure 5.10 The five configurations (symbols) of the EdgeBreaker algorithm. (Maglo et al. 2015)

Basically, the compression is traversing the dual graph of mesh in depth-first order. The current loop is split into two when the split (S) case is met. One of them (left-part) is pushed into the stack. The other (right) is further traced. Later, every loop in the stack is also traced and encoded.

The EdgeBreaker method able to encode the connectivity of orientable manifold meshes. These meshes may have multiple boundaries or have arbitrary genus. The best part of EdgeBreaker is it guarantee a worst-case cost (upper bound) for simple meshes. However, the original EdgeBreaker algorithm is incompatible with streaming applications, because it needs a two-pass process for decompression. A disadvantage of an EdgeBreaker is that it requires about the same bitrate for non-regular and regular meshes.

The decoding efficiency was also improved to overcome linear time and space complexities in (King and Rossignac 1999, Rossignac and Szymczak 1999, Isenburg and Snoeyink 2001) of EdgeBreaker algorithm.

The Angle Analyzer is a geometry-driven mesh traversal single-rate compression algorithm developed by (Lee et al. 2002). The EdgeBreaker algorithm's five descriptors and the traversal design for minimizing the entropy revisited in the Angle-Analyzer algorithm. The connectivity encoding is also performed by a gate-based approach with cooperation between geometry and connectivity, in order to achieve an efficient mesh traversal driven by both criteria adaptively.

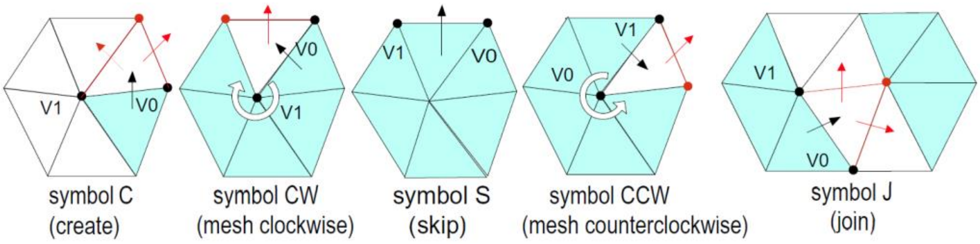


Figure 5.11 Angle-Analyzer set of symbols (Lee et al. 2002)

In Figure 5.11 the red vertices are front ones, red gates are new gates to be inserted into the gate list to continue to conquest. Create (C) symbol is generated, if the front vertex has not been visited yet. In the ordered gate list, two new gates replace the current gate, as in the original EdgeBreaker. If the front vertex has been previously visited, the front vertex can be located by turning either clockwise (CW) around V1 or counter-clockwise (CCW) around V0 when the front vertex has been visited and a new gate will replace both the current gate and the next gate in the list. When an active gate updated to mesh boundary, there is no front face and a symbol skip (S) will be generated. When the decoder is not able to identify the location of a previous front vertex, a symbol J will be generated followed by an offset.

One of the strengths of this algorithm is its quadrilateral mesh processing implementation. The algorithm is generalized to quadrilaterals so that the Angle-Analyzer is not only able to handle triangle meshes but also quadrilateral meshes with extended the triangle conquest approach.

To minimize the entropy of the op-code sequence, the Angle-Analyzer chooses next gate adaptively to dominate the splitting and the merging of edge loops.

5.1.4. Valence Encoding

The number of triangles is doubled the number of vertices in the manifold mesh. That means if an algorithm implements its connectivity compression on triangle-based approach, it will have twice as much output as vertices-based approaches. Therefore one symbol per vertex will lead to better connectivity compression performance. Representing connectivity with one symbol per vertex established with valence approach.

Euler’s theorem pointed out that the average vertex valence is 6. In fact, valence distribution is focused around 6 in most models. To exploit these statistics, valence-based encoding algorithms are developed.

The pioneering valence-driven approach has developed by (Touma and Gotsman 1998). This approach (TG98) doesn’t have a new traversal method.

Traversal method and order is the same with EdgeBreaker. Even the behavior of algorithm for configuration of symbol C and configuration of symbol S is the same in TG98 algorithm. However, TG98 algorithm treat R, L and E symbols as in EdgeBreaker differently. Instead of encoding L, R, and E, TG98 encode the valence of inserted vertices which is generally around 6. When a configuration of S occurs TG98 encode the offset for each S triangle. Concluding the last element of a triangle fan, TG98 completed automatically with the missing L, R, or E triangle valence information.

The generated list of vertex valences is ready to be compressed by an entropy coder effectively because valences are mostly close to each other difference is relatively small. Special cases like splitting the current active loop or merging it with another active loop. These cases are covered with special codes in the encoding phase. In order to make 3D mesh model 2-manifold and closed topology a dummy vertex is added and connected to all boundary vertices.

An example run of TG98 encoding algorithm stated in Figure 5.12. The active lists are marked by thick lines. Edges already traversed are dashed lines. (a) Input mesh. (b) Dummy vertex added and connected to all boundary vertices. (c) Pick initial triangle to start, mark focus vertex, and generate code words “add 6, add 7, add 4”. (d) Expand the active list and generate code word “add 4”. (e) “add 8”. (f) “add 5”. (g) “add 5”. Focus vertex becomes full (all edges encoded). (h) Focus vertex removed, and focus moved on along the active list. (i) “add 4”. (j) “add 5”. Now the next free edge of the focus leads to a vertex already in the active list. (k) Active list split into two. Generate code word “split 5” (5 is the offset), and smaller one pushed on the stack. (l) Focus vertex removed, and focus moved on. (m) “add 4”. (n) “add 4”. Focus vertex is full so it is removed. (o) The dummy vertex is added “add dummy 6”. (p) First active list complete. The second active list popped from the stack. (q) “add 4”. (r) Focus vertex removed, and focus moved on. (s) Focus vertex removed, and focus moved on. (t) Second active list complete. The resulting code is “add 6, add 7, add 4, add 4, add 8, add 5, add 5, add 4, add 5, add 5, add 4, add 4, add dummy 6, add 4".

TG98 is a powerful compression algorithm when combined with the context-aware arithmetic encoder. However, this algorithm can only work with orientable manifold meshes.

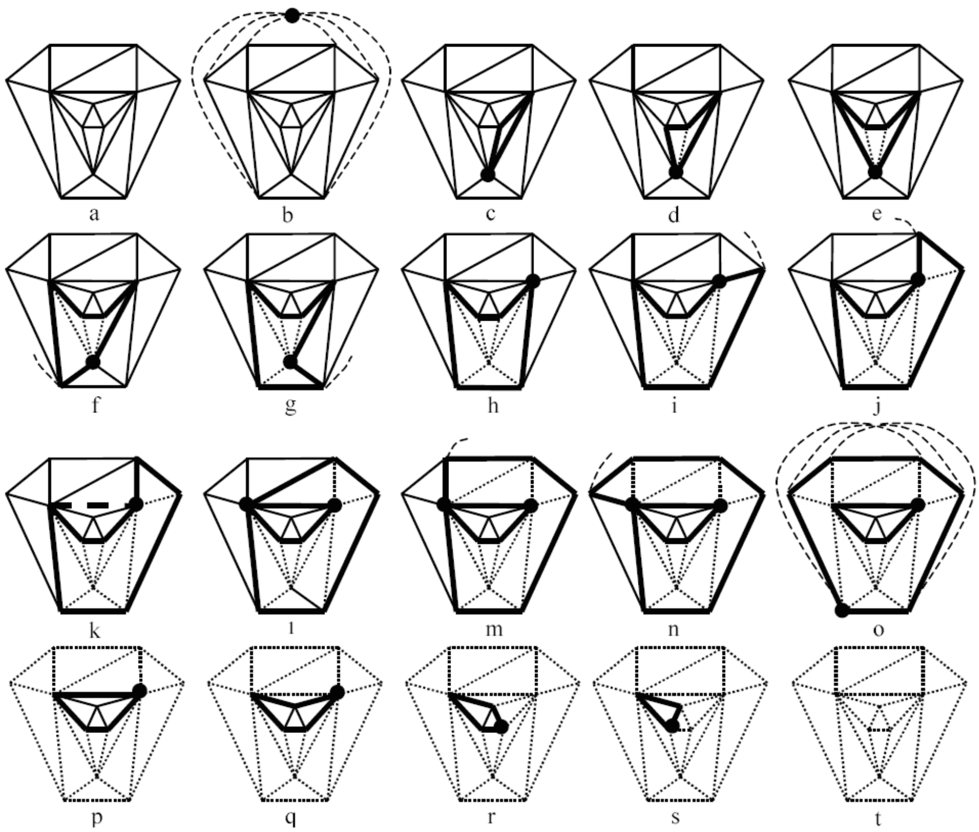


Figure 5.12 Example Run of the (Touma and Gotsman 1998) encoding algorithm

(Alliez and Desbrun 2001b) observed that split operations and dummy vertices are not too little to ignore as stated in TG98 algorithm. They propose to replace the deterministic conquest by a heuristic method to which is proved to be better at choosing the next focus vertex with the minimal number of free edges. Split codes and even the range of split offsets are minimized by this replacement. It also improves compression rates when objects have numerous boundaries. Valence-driven connectivity encoding stated the first upper bounds in valence-based approaches, confirming the correctness of a valence-driven algorithm. In addition, they encoded the output symbols with the range encoder (Schindler 1998), an effective adaptive arithmetic encoder.

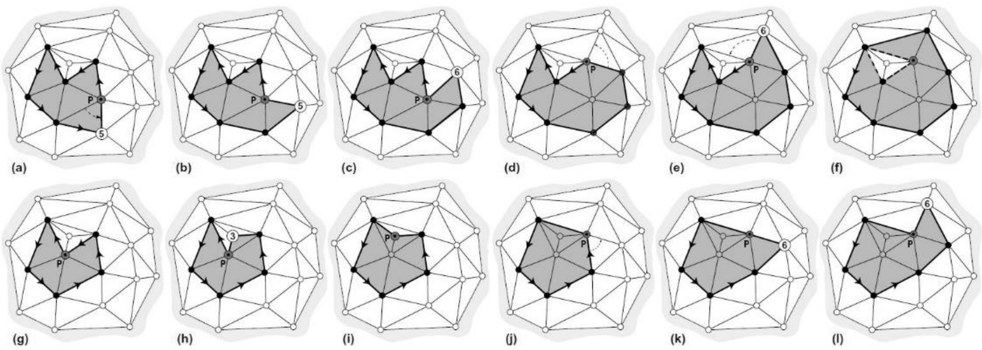


Figure 5.13 Top line the original algorithm (Touma and Gotsman 1998), Bottom line (Alliez and Desbrun 2001a)

Comparison of traversal methods for both valence-driven connectivity and TG98 in Figure 5.13: Above demonstration is based on TG98. (a) The next counter-clockwise edge is conquered from the active pivot. A valence code 5 is output. (b) code 5. (c) code 6. (d) the pivot, full, is removed from the list at no cost. The next vertex in the active list is chosen as a pivot. (e) code 6. (f) code split with an offset of 2. The code sequence is {5,5,6,6,split(2)}. Below demonstration is based on valence-driven connectivity method. (g) the best pivot candidate is searched into the active list. One unique vertex has only one free edge, it is thus chosen as a pivot. (h) code 3. (i) the full pivot is removed. The best pivot candidate has 0 free edges. (j) the full pivot is removed. The next best pivot candidate has 2 free edges. (k) code 6. (l) code 6. The pivot is now full. The code sequence is {3,6,6}.

(Alliez and Desbrun 2001a) they claimed to have demonstrated the optimality of valence-based approaches. On the other hand, this algorithm can only work with orientable manifold meshes as like TG98 algorithm.

The FreeLence (name comes from the free valence) encoder implement a different approach which is counting the number of unconquered edges adjacent to the processed vertex, except for split cases (Kälberer et al. 2005). As in full-valence coders, the number of symbols is closely related to the total number of vertices, so that their code sequences and symbol dispersions can be seen as in Figure 5.14.

Using free valences heavily dependent on traversal algorithm. It needs to be combined with an appropriate traversal algorithm. Because of that, FreeLence employs a geometry-driven traversal scheme to keep the active list as convex as possible, like in the Angle Analyzer encoder (Lee et al. 2002).

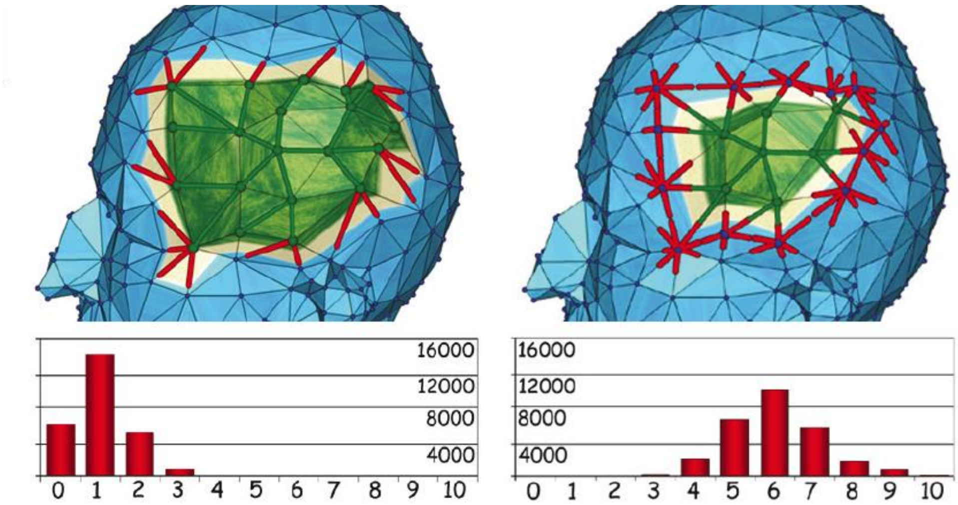


Figure 5.14 Geometry-driven coding with free valences (left) will in practice yield a lower symbol dispersion than coding with full valences (right) (Kälberer et al. 2005)

(Mamou et al. 2009) proposed a distinctive valence approach which has the capability to encode non-manifold and non-oriented triangle meshes, called TFAN

(Triangle Fan-based compression). It partitions the mesh into a set of predefined triangle fans. TFAN partitions a triangle mesh into a set of predefined triangle fans as a preprocessor step. For each triangle fan, there needs to be configuration code and degree of the fan. There are 10 predefined configurations Figure 5.15 that can cover all connectivity information with the help of its degree, regardless of orientation or being manifold.

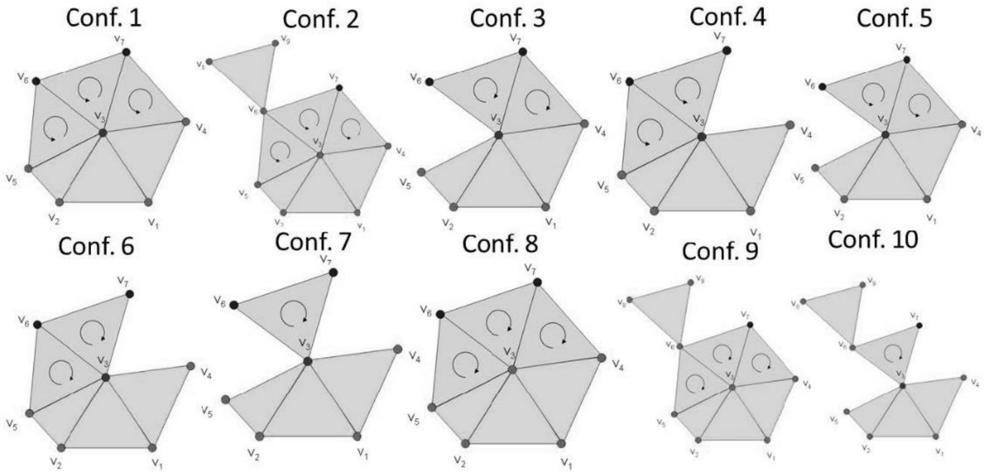


Figure 5.15 The ten TFAN configurations (Mamou et al. 2009)

TFAN has been successfully implemented and be a part of MPEG4/3DGC and Open3DGC (Mamou 2009) methods.

5.2. Geometry Compression

3D geometry information of a mesh is an elephant in the compression room. Most of the early work try to reduce the size of connectivity information and not even try to compress geometry at all. That why geometry information occupies considerably more storage space than connectivity information in almost all cases because connectivity compression has been studied extensively.

Usually, geometry compression steps are similar like quantizing the vertex location first and then vertex position prediction. Accurate prediction outputs a small prediction error. Storing this small prediction errors with the help of delta difference make the file suitable for better compression with entropy encoders.

5.2.1. Quantization

Geometry data of vertex coordinates are often stored in precise 3 (x, y, z) IEEE 32-bit floating point values and thus consume quite an important part of the whole 3D data. Also, geometry compression is challenging because it deals with floating point numbers rather than integers as in connectivity compression. The 8-bit exponent of 32-bit IEEE floating-point numbers allows positioning of the known universe: from billions of light years, down to the sub-atomic particles. That much precision is not needed mostly. Reducing precision by applying quantization can significantly lessen data size without perceivable quality loss. Some applications accept some precision loss in order to achieve superior compression rates. (Oral and Elmas 2017)

5.2.1.1. Scalar Quantization

Scalar quantization is transforming number representation of vertex from the floating-point into an integer which is also called normalization. The bounding box of the mesh partitioned into a grid in 3D. Bounds of the grid are found by the biggest number can be codded with that quantization bits amount. Cell size can be either uniform or non-uniform. Center of a cell represents each vertex that is within the bound of that cell. Positions are generated from the 3 coordinates of the cell.
Well-known compression schemes use a uniform scalar quantization (Deering 1995a, Taubin and Rossignac 1998, Touma and Gotsman 1998, Rossignac 1999).

Unlike connectivity, geometry is slightly altered after quantization. Quantization is not a lossless application. It affects the results in an irreversible manner. However, the amount of impact on compression size is non-trivial.

Angels can also be used to represent geometry information. (Bajaj et al. 1999) and Angle-Analyzer (Lee et al. 2002) encode the geometry information with three angles. The Angle-Analyzer store only one dihedral angle and two internal angles as a geometry. There is also a quantization step too. Quantization not applied to global coordinates but applied to local angels. Different quantization bits applied to the different angles, in order to achieve better rate-distortion performance.

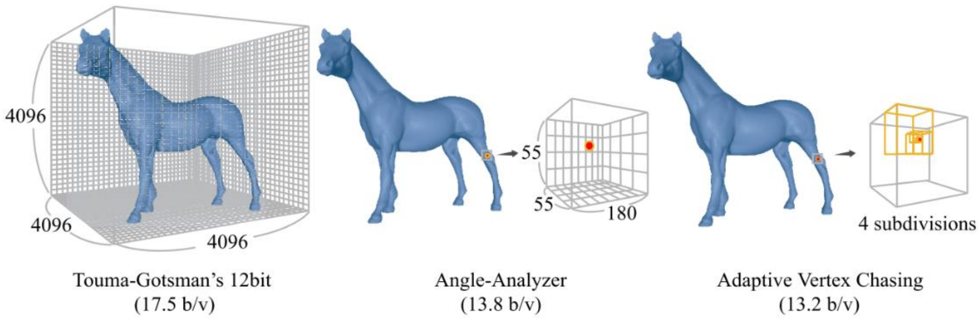


Figure 5.16 TG98 uniform quantization, Angle-Analyzer non-uniform quantization, Adaptive Vertex Chasing 4 subdivision (Lee and Park 2005).

(Lee and Park 2005) proposed locating the geometry coordinates within four different ranges. The biggest range is relatively having few vertices than others. To encode the coordinates within a range, the ranges are roughly subdivided depending on their size Figure 5.16. The position of the vertex is encoded by the range type and the sub-cell number.

5.2.1.2. Vector Quantization

Another alternative quantization method is Vector Quantization. Like scalar quantization but the shape of a cell that scalar quantization has is cube or cuboid. However, in vector quantization shape can be arbitrary that has grouped vertex locations in it that divides the set of points to quantize into arbitrary-shaped groups.

Vector quantization has been proposed for geometry compression in (Lee and Ko 2000, Chou and Meng 2002). Vector Quantization does not follow the usual quantization-prediction-entropy coding approach. In contrast, the vector quantization approach first predicts vertex positions and then compresses the three components of each prediction residual together. Quantization cells are not cube anymore. Their shape can better adjust to the component or model shape. Each group has a single point to represent themselves. Codebook needs to be transmitted with the model's compressed data. Vector quantization has demonstrated (Lee and Ko 2000, Chou and Meng 2002, Bayazıt et al. 2007, Lu and Li 2008, Meng et al. 2010) its ability to achieve better rate-distortion performance other techniques. However, the determination of the cell shape is computationally hard.

5.2.2. Prediction

Quantization of vertex coordinates followed by a prediction of vertex positions. Prediction is based on guessing the relation between adjacent vertices' coordinates. The fundamental point of prediction methods is that it reduces the amount of geometry data. If the quantization results occur within the small range that means predictions are good and many of the coordinates of the corrective vectors will be small integers. A variable length or entropy coding schemes replace the frequently appearing integers with shorter codes. Thus, in highly skewed models on the quantization phase, compression performance depends heavily on the precision of the vertex estimates.

Several prediction schemes have been proposed. such as delta prediction (Deering 1995a, Chow 1997) used delta prediction. (Taubin and Rossignac 1998) used linear prediction. (Touma and Gotsman 1998) used parallelogram prediction. (Bajaj et al. 1999) used second-order prediction. All these prediction schemes can be treated as a special case of the linear prediction scheme.

The position of the next vertex is predicted to be the position of the previous vertex. This method is called delta prediction (Deering 1995a, Chow 1997). The two positions, the delta, is encoded. (Bajaj et al. 1999) use a second-order predictor. The differences between following deltas encoded by the second-order predictor. The linear prediction has been used on the Topological Surgery algorithm (Taubin and

Rossignac 1998) which uses linear prediction. Prediction is done by the linear combination of the k previous vertices in the vertex spanning tree.

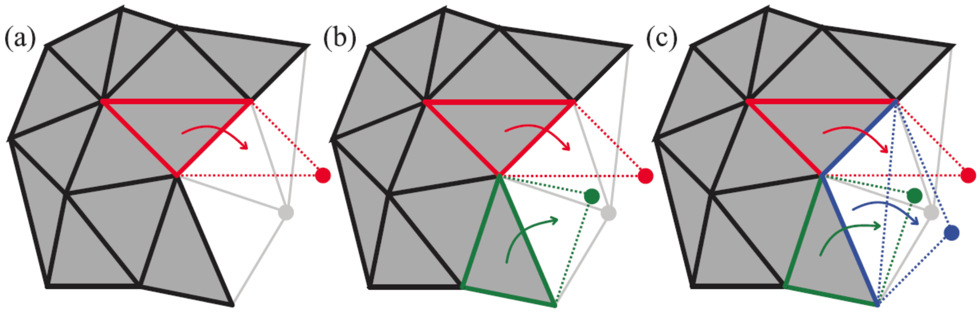


Figure 5.17 Simple (a), Dual (b), FreeLence Parallelogram Prediction (c) (Maglo et al. 2015)

Parallelogram prediction has been introduced by (Touma and Gotsman 1998), a founding idea that has spawned numerous descendants, besides their ground-breaking valence-driven connectivity encoding. The compression algorithm starts with a new vertex and with a triangle from an edge. A new vertex has been predicted and the position of that prediction creates a parallelogram with two edges and one vertex. Figure 5.17(a). The average position created from two parallelograms stated the dual parallelogram prediction. Figure 5.17(b). 75% of cases are appropriate to use dual parallelogram prediction. (Sim et al. 2003). It provides slight improvements over parallelogram prediction. Combination of three parallelogram prediction has been used by The FreeLence coder (Kälberer et al. 2005). Two of them is standard parallelogram prediction, the third one is from the joining of two outer virtual edges. Figure 5.17(c).

5.3. Entropy Coding

Data compression is mainly based on two approaches in the modern era. One of them is Huffman Coding, the other is Arithmetic Coding. Arithmetic coding can also be named as Range coding. Unlike Arithmetic coding Huffman coding is much faster. Arithmetic coding can easily approach theoretical compression limit which is defined as Shannon entropy, but arithmetic coding is a computationally heavy algorithm.

Asymmetric numeral systems (ANS) and Asymmetric binary systems (ABS) by (Duda 2013) is a fairly new approach to entropy coding, which allows ending this tradeoff between speed and rate. ANS is approximately 50% faster decoding than Huffman coding. The compression rate of ANS is almost similar to arithmetic coding even sometimes better than arithmetic coding.

Various algorithms were discussed in this thesis. These algorithms often use Huffman and Arithmetic coding methods. To be precise (Deering 1995a, Chow 1997, Bajaj et al. 1998, Gumhold and Straßer 1998, Taubin and Rossignac 1998, Touma and Gotsman 1998) use Huffman coding in their methods either in connectivity or geometry compression part. (Isenburg and Snoeyink 2000, Alliez and Desbrun 2001b, Lee et al. 2002, Diaz-Gutierrez et al. 2005, Kälberer et al. 2005, Lewiner et al. 2006, Mamou et al. 2009, Buelow et al. 2017) use various arithmetic/range coding methods: order-0 to order-3, and Range Encoder (Schindler 1998) which is an adaptive arithmetic encoder. Only (Ponchio and Dellepiane 2015) use Tunstall coding. Unlike Huffman coding and other variable-length schemes, Tunstall coding (Tunstall 1967) use a fixed number of bits from a variable number of symbols. In the decompression phase, the input blocks consist of a fixed number of bits and the output is a variable number of symbols, Tunstall code performance on compression almost equal with Huffman, especially where the bit size of the input block is small. The EdgeBreaker algorithm didn’t use any entropy coder. Their results do not depend on entropy or arithmetic coding schemes. Therefore, EdgeBreaker is proper for compressing every kind of models. Especially attractive for compressing large datasets which include lots of small models.

6. EXPERIMENTAL DESIGN

This chapter covers the work we have done throughout the implementations and comparisons as well as obstacles that we have encountered and resolved.

Some algorithms which are mentioned in chapter 5 or in survey papers (Taubin and Rossignac 1999, Alliez and Gotsman 2005, Peng et al. 2005, Maglo et al. 2015), are not available to end-user or not even published at all. Reaching authors for every method that is not publicly available is not the case. Implementing from their paper may resolve the problem but while coding original intentions might not be maintained. Different implementations may reveal different programs which are not reliable for using comparison purposes. Some popular algorithms don't have a compiled version or outdated development environment requirement.

At first prior 3D mesh compression methods intended to compare with each other. We have started with Deering's Geometry Compression algorithm which is acquired by Java3D. However, Java3D is no longer supported from Java community but still publicly available through various links. (Deering 1995b) A demo implementation was found. The algorithm could not be compiled with the current Java versions. Java3D was last updated with Java 1.5. With this information, we have created a development environment with Java 1.5 and Java3D for compiling the existing demo implementation. Deering's demo software was compiled and run successfully which gives a single binary output file.

EdgeBreaker is an easily found publicly available algorithm on the Internet. Rossignac shared the first version of EdgeBreaker with the public. However, version shared by Rossignac uses a special data structure called Corner Table. In order to use the EdgeBreaker, it is necessary to convert the raw model data (OBJ or OFF) to the Corner Table data structure (OV Table). Part of an online implementation of EdgeBreaker does have a converter for OFF file to OV file called OFF2OV. We have implemented that library and create the OV table of each model. The EdgeBreaker can then compress the connectivity of 3D models with this OV Table accordingly.

After the compression, there is connectivity information (CLERS) and reordered geometry information in separate files. There also exists a modified version of EdgeBreaker on the Internet. Google Draco is also using EdgeBreaker as a base method for one of their compression levels. (Brettle and Galligan 2017) EdgeBreaker's versions have been tried but full control over quantization couldn't be established. That's why the original version is used in this thesis.

The valence-driven connectivity encoding method of (Alliez and Desbrun 2001a) published as a closed source application which is part of 3D Toolbox of Pierre Alliez (Alliez and Desbrun 2001b). 3D Toolbox can compress 3D mesh files with the valence-driven connectivity encoding method. However, it only compresses connectivity and outputs text and a binary version of connectivity information. Geometry information can be compressed with 3D Toolbox which can only output binary version but the used method is not defined or named well. (Touma and Gotsman 1998) was one of the pioneering algorithms but source code or implementation is not publicly available. Valence-driven connectivity encoding algorithm has improved the original TG98 algorithm, which is not needed. Valence-driven connectivity encoding algorithm has a strong geometry compressor in its published paper but we don't have a chance to use it because 3D Toolbox doesn't define its methods clearly.

Although polygonal mesh compression is not a topic of this thesis, (Isenburg and Snoeyink 2000) developed a Face Fixer method to apply 3D mesh compression directly to polygons. Later (Isenburg 2000) has specialized this Face Fixer to triangles and published Triangle Fixer method but didn't release any implementation other than Face Fixer. This method creates Face Fixer labels as connectivity information. Connectivity information extracted with Face Fixer was also compressed with selected general-purpose algorithms. Since our dataset only contains triangulated models, Face Fixer which is developed specifically for polygons, shouldn't be compared with other triangle compression methods. For information purpose only, in the results chapter, we have also shown the performance of Face Fixer connectivity coder too.

(Mamou 2009) developed and published the TFAN algorithm with Open3DGC as its implementation with support from AMD (rest3D). Published demo implementation was easy to compile and run. The Open3DGC algorithm treats the file as a whole and compresses the connectivity and geometry information together. The result is a single binary file.

(Chun 2011) developed and released a WebGL-Loader for a Google project called Google Body. Although this project has been shelved by Google, it has been included in the thesis because it has been cited by other articles and included in the comparisons. Finding correct branch of WebGL-Loader was not easy. There are people who try to continue this project by themselves and there is the outdated original version. POSIX implementation of algorithm compiled by the help of emulators.

An extension to the Cut-Border Machine which compresses both the connectivity and attributes without loss has been implemented and released named as Harry (Buelow et al. 2017).

Implementation of Polygon Mesh Compressor on the web, which is not working anymore due to browsers intentional lack of Java support, and also a standalone Java version has been released by (Isenburg et al. 2002). This benchmark software covers (Touma and Gotsman 1998, Alliez and Desbrun 2001a, Isenburg 2002, Isenburg and Alliez 2002, Khodakovsky et al. 2002) methods. Therefore, lots of methods interleaved with each other. We can’t use this implementation.

In 2008, ISO/IEC 14496-25 standard for 3D Model Compression was adopted by the (MPEG) Moving Picture Experts Group, referred to as MPEG-4 Part 25, 3D Graphics Compression Model. (Jovanova et al. 2008) Implementation of this Part 25 and also Part 16 software of MPEG-4 Standard have been released by (Preda 2008). However, there is only one reference software which covers lots of standards and couldn’t compile successfully. Group patented the MPEG-4, therefore there isn’t any publicly available implementation of MPEG-4. One of the sub-algorithms of MPEG-4 is Topological Surgery which developed and released in the IBM research center but IBM has changed its structure and this project is also not available anymore even the internet archive doesn’t have the binaries.

Apart from the above methods, dealt with the hassle of compilation or even finding, there are applications, frameworks, or opensource programs which can be easily compiled or downloaded too: OpenSceneGraph (Osfield 2001), Extensible 3D format, OpenCTM (Geelnard 2009), Draco (Brettle and Galligan 2017), and Corto (Ponchio 2015).

Open source project OpenCTM is a file format and at the same time a software library and a compression toolset of 3D meshes which has three types of codecs: RAW, MG1(lossless), MG2. (Geelnard 2009) OpenCTM is a lossless format. OpenCTM have different compression methods for different needs. MG1 and MG2 are lossless compression methods which utilize triangle reordering and apply LZMA to compress the connectivity information. MG1 and MG2 differentiate at the floating-point storage method. MG1 store original vertex data as floating-point on the other hand MG2 store fixed place values. Heavily compression comes from the lossless prediction technique. LZMA coder can handle small data very well so the prediction technique of MG2 tries to minimize value range of the vertex coordinates and also attributes too.

Corto is a library for compression and decompression of meshes and point-clouds. The main focus of Corto is decompression speed, while still providing good compression rates. (Ponchio 2015)

Draco is an open-source library for compressing and decompressing 3D mesh data, developed by Google Chrome Media team (Brettle and Galligan 2017) which has different compression levels. One of the methods Draco has implemented is EdgeBreaker with rANS.

6.1. General-Purpose Data Compression Methods

TurboBench (Bouzidi 2013) software includes almost all popular, latest or fastest compressors in one compiled software package. All of the raw models and (Rossignac 1999, Isenburg and Snoeyink 2000, Alliez and Desbrun 2001a) connectivity, and geometry data have been benchmarked with TurboBench software (Oct 7, 2018) and below ten best compressors according to the results have been chosen to be compared.

Bcm (Muravyov 2008) is a high-performance file compressor that utilizes advanced context modeling techniques to achieve a very high compression ratio.

Zpaq (Mahoney 2009) is a free and open source incremental, journaling compressor. The compression algorithm uses an optional bitwise context mixing model, followed by arithmetic decoding, packing into bytes, and an optional post-processing transform.

Bzip2 (Seward 1996) is a free and open-source file compression program that uses the Burrows-Wheeler algorithm. The program is more effective than Deflate and LZW programs. The LZW or .z and the Deflate algorithms such as .gz and .zip are less effective but they operate quickly. As a result, they end up taking more space than what bzip2 can achieve. Bzip2 relies on Burrows-Wheeler transform or algorithm to convert all character sequences recurring frequently into identical letters strings. The program then uses the Huffman coding move to front transform. bzip, which was the predecessor of bzip2, employed arithmetic coding but the successor uses Huffman coding. The performance of bzip2 is asymmetric. It has a relatively fast decompression.

Lzlib (Diaz 2009) is a data compression library providing in-memory LZMA compression and decompression functions, including integrity checking of the decompressed data. The high compression of LZMA comes from combining two basic, well-proven compression ideas: sliding dictionaries (LZ77/78) and Markov Models (the thing used by every compression algorithm that uses a range encoder or similar order-0 entropy coder as its last stage) with segregation of contexts according to what the bits are used for.

Lzma (Lempel-Ziv Markov Algorithm) LZMA is a compression format invented by Igor Pavlov, which combines an LZ77 compression and range encoding. LZMA uses a dictionary compression algorithm (a variant of LZ77 with huge dictionary sizes and special support for repeatedly used match distances), whose output is then encoded with a range encoder, using a complex model to make a probability prediction of each bit.

Zstd or Zstandard (Collet 2015), is a fast lossless compression algorithm, targeting real-time compression scenarios at zlib-level and better compression ratios. It's backed by a very fast entropy stage, provided by Huffman coding and Finite State Entropy (Collet 2013) library.

Balz (Muravyov 2016) uses LZ77 with arithmetic coding, a 512K buffer with optimal parsing (Storer and Szymanski 1982)

Brotli is a generic-purpose lossless compression algorithm by (Google 2015) that compresses data using a combination of a modern variant of the LZ77 algorithm, Huffman coding, and 2nd order context modeling, with a compression ratio comparable to the best currently available general-purpose compression methods. It is similar in speed with deflate but offers more dense compression.

Lzham (Geldreich 2009) is short for LZMA-Huffman-Arithmetic-Markov. It is based on LZMA (7zip) but instead of using arithmetic coding throughout, it uses them only for binary decisions and uses Huffman or Polar codes for literal and match codes. A Polar code is similar to a Huffman code but is simpler to calculate at a cost of 0.1% in compression.

Libdeflate is a library for fast, whole-buffer DEFLATE-based compression and decompression. libdeflate is heavily optimized. It is significantly faster than the zlib library, both for compression and decompression and especially on x86 processors. In addition, libdeflate provides optional high compression modes that provide a better compression ratio than the zlib's "level 9".

Table 6.1 Selected 10 compressors listed based on compression algorithms

Based on Compression Algorithm	Compressors	Extra Applied Methods
Burrows-Wheelers Transform	bcm	Order-0 Context Modeling
	bzip2	
Reduced Offset Lempel-Ziv	balz 1	Arithmetic Coding
Lempel-Ziv-Markov-Algorithm	lzlib 9	
	lzma 9	
	lzham 4	
Lempel-Ziv-77	zpaq 5	Heavy Context Modeling
	brotli 11	Order-2 Context Modeling
	zstd 22	Finite State Entropy
	libdeflate 12	

A new configuration for TurboBench software has been created for above 10 general-purpose compressors. Connectivity and geometry information separately benchmarked with this new configuration.

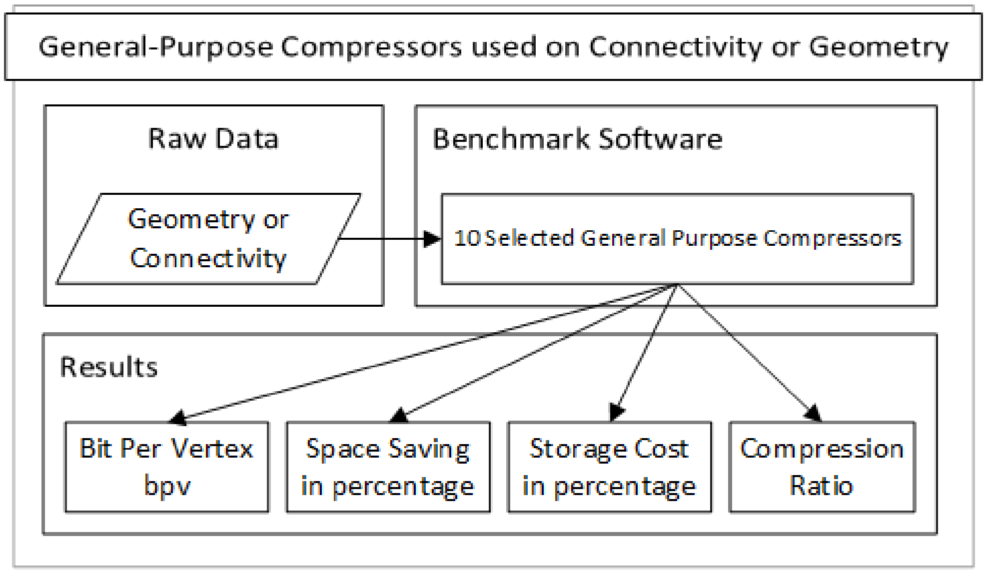


Figure 6.1 Connectivity or Geometry information benchmark scheme

6.2. Dataset

Our dataset has been created with the models collected from public domain. We have tried to use similar models from other datasets (Turk and Levoy 1996, Turk and Mullins 2000, Desbrun 2004). Our dataset consists of 20 folders for each model. Each folder consists of 25 different files representing the result of 17 different encoding methods with their binary version if available.

All models are selected among the oriented manifold meshes. Models were selected accordingly to test the limits of the methods. For example, there is the Statue model which contains 1 million vertices and there is Geosphere model which contains only 162 vertices to test with a high and low number of vertices.

There exists model which has holes or boundaries. Our collected methods can handle boundaries and holes.

One connected component applied Euler formula has been studied by researchers generally. Therefore, all of the models consist of only one connected component, for example, the implemented original EdgeBreaker method can only process one connected component.

Some models, especially Happy Buddha, include duplicated vertices. According to the data gathered from TriMeshInfo, some models have self-intersection. A different number of genus options are available generally 0 but throughout the dataset, genus can be from 0 to 104.

Model images were taken with the help of MeshLab software (Cignoni et al. 2008). General mesh info extracted with TriMeshInfo software (Cignoni et al. 2005).

Since connectivity information implicitly stored, raw file formats can be a different size. Therefore, the base file format can be OBJ, OFF, OSG, PLY etc. In our study, we have selected the OBJ file format as a base file format to show the compression performances of collected methods.

6.3. Design of Our Approaches

As stated earlier some algorithms process connectivity and geometry separately which make it possible to look for a better general compression algorithm for only connectivity information or only geometry information. In this thesis, EdgeBreaker CLERS connectivity data, Face Fixer labels data, and the valence-driven connectivity encoding data have been gathered for all models. Geometry information has been gathered from original EdgeBreaker algorithm Vertices text output, which was just sorted geometry information.

Beginning of some sample connectivity information of Body model given at Table 6.2 in order to visualize the data.

Table 6.2 Connectivity information samples from Body model

EdgeBreaker	Valence-Driven Con.	Face Fixer
CCCRCRCCCRCCRCC	6567767597485565955	F3F3F3F3F3F3RF3F3R
CRCCRCCCRCRCCCC	86566657546858676575	F3F3F3F3F3F3RF3F3F
RRLCCRCRRCRCCCC	67676568675757646676	3F3RF3F3F3F3RH24RF
RCRCRRCCCCR…….	665686865565657…….	3F3RF3F3RF3F3………

Now we have two connectivity and one geometry file to test general-purpose data compressors compression performance against other 3D mesh compression methods. We have tried two approaches to creating a complete model. In this thesis, we have used the total model term which is a complete model with a general-purpose compressor(s) applied.

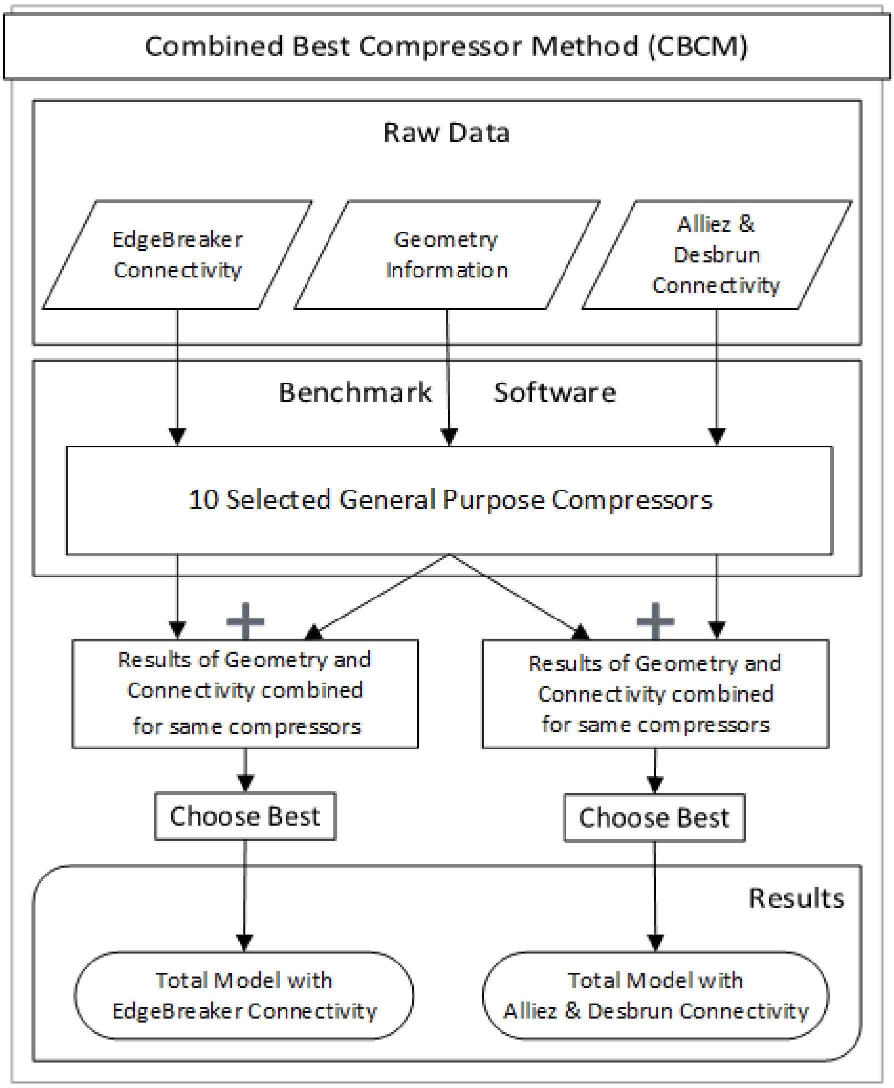


Figure 6.2 Combined Best Compressor Method (CBCM)

Our first approach is combining results of connectivity and geometry data before choosing the best compressor Figure 6.2. We have named this approach as Combined Best Compressor Method (CBCM). CBCM is appropriate for those who don't want to use more than one general-purpose compressor. In the end, CBCM decides for a single compressor which is easy to use for a complete model.

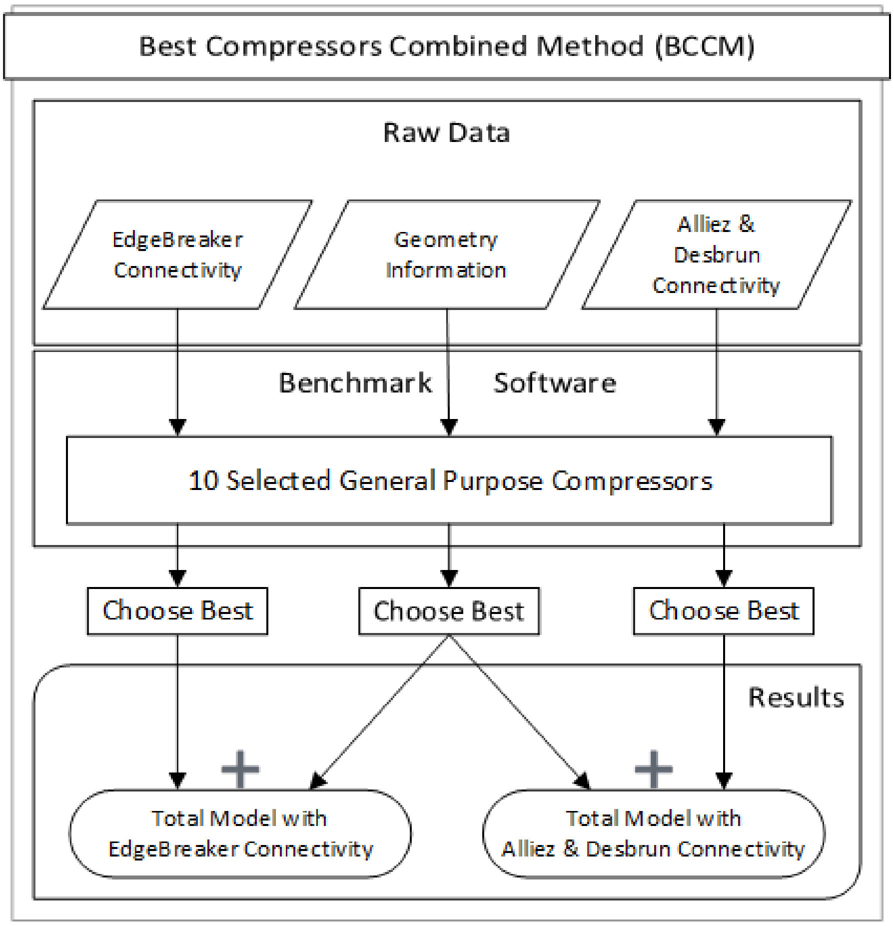


Figure 6.3 Best Compressors Combined Method (BCCM)

Our second approach is choosing the best compressor for connectivity and geometry separately before combining their results to a single file Figure 6.3. We have named this approach as Best Compressors Combined Method (BCCM). BCCM is appropriate for those who need better compression. It may use one compressor according to model but experimental results show that generally, it uses two compressors because geometry and connectivity data has often different better compressors.

6.4. Collected Methods and Final Testbed

All the collected methods have been analyzed for comparison purposes. Some of the methods are forcing the user to quantize the mesh, some of the methods are calculating normal information and adding into the data without asking. In order to ensure the standard between methods, elimination has been done among them. Finally, there are only 10 methods left to compare against each other in a reliable way. Two of them, (Rossignac 1999, Alliez and Desbrun 2001a) which are allowing interruption at the last stage, have been combined with our two approaches (CBCM and BCCM) at the general-purpose compression stage.

We have designed a testbed for our experimental design Figure 6.4 which accepts input as OBJ or OFF raw format. After applying each of 3D mesh compression method to each of 20 uncompressed model data, performance results are represented with four different way: bit per vertex, space saving that method can provide in percentage, storage cost after compression according to raw data in percentage too and finally the compression ratio.

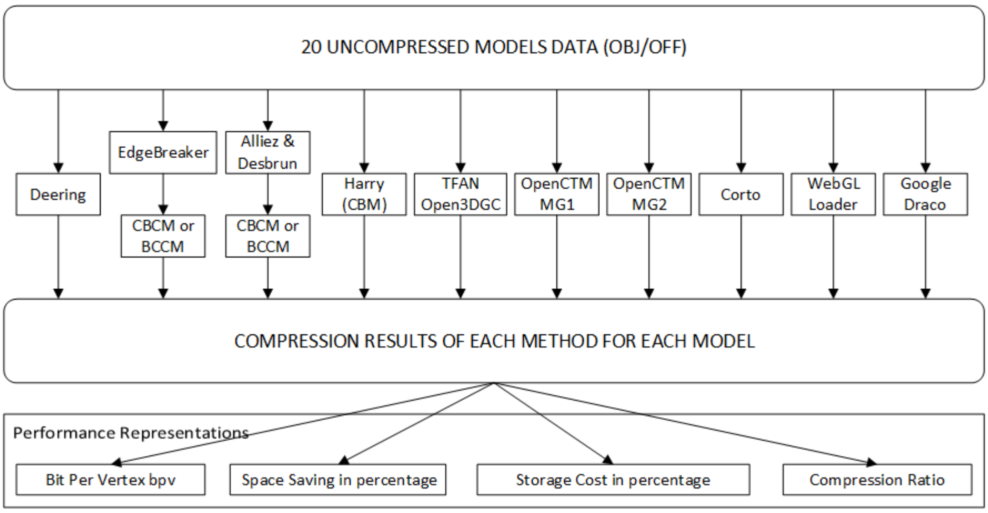


Figure 6.4 Final Testbed of our design for the comparison test

7. RESULTS AND DISCUSSIONS

There are three types of results in this chapter. One of them is a performance result of methods in chapter 4.1. The second one is general-purpose compressors' performance on 3D model data. The third one is a comparison of collected 3D mesh compression methods while applying current best general-purpose compressors also with our two approaches.

Benchmark results are supported with the ranking algorithm in order to generalize the compression method among models.

7.1. 3D Mesh Compression Methods

A complete summary of chapter 4.1 with categorically sorted and storage cost given experimentally or by reference is given at Table 7.1.

Table 7.1 Connectivity compression rates of prior algorithms categorically

Category	Algorithm	Storage Cost
Generalized Triangle Strip	(Deering 1995a)	1:4 - 1: 10 / 8-11 bpv
	(Chow 1997)	1:30 - 1:37
Spanning Tree	(Taubin and Rossignac 1998)	2.48 - 7 bpv
	(Diaz-Gutierrez et al. 2005)	2* bpt
	(Li and Kuo 1998)	1,5 bpt
Layered Decomp.	(Bajaj et al. 1999)	1.4 - 6.08 bpv
Valence-Driven	(Touma and Gotsman 1998)	0.2 - 2.4 bpv
	(Alliez and Desbrun 2001a)	0.024 - 2.96 /3.24* bpv
	(Isenburg and Snoeyink 2000)	1.67 - 2.92 bpv
	(Kälberer et al. 2005)	0.03 - 2.11 bpv
	(Mamou et al. 2009)	0.2 - 2.7 bpv
Triangle Conquest	(Gumhold and Straßer 1998)	3.22 - 8.94 bpv
	(Gumhold 1999)	0.3 - 2.7 bpv
	(Rossignac 1999)	1.8 - 2.4 / 4* bpv
	(King and Rossignac 1999)	3.67* bpv
	(Gumhold 2000)	3.52* bpv
	(Szymczak et al. 2001)	1.622* bpv
	(Lee et al. 2002)	1.5 / 4* bpv

*Theoretical upper bounds for connectivity given according to their articles.

Although geometry compression of (Deering 1995a)method is a lossy method it is one of the first work in 3D mesh compression. Deering didn't state a bit per vertex performance values rather gave a compression rate of 1:4 to 1:10. Later other researchers have demonstrated its application and acquired 8 to 11 bpv performance values. Geometry Compression of Deering specifically developed for hardware implementations. Its main purpose is reducing the memory usage between CPU and GPU. However, Deering didn't state any decomposition in its original paper. Later (Chow 1997) implemented the generalized triangle mesh's decomposition and inspired by the Topological Surgery algorithm. Performance value of Chow's paper is also showing a compression rate of 1:30 to 1:37. We couldn't find any implementation or experimental results about bpv value.

Topological Surgery algorithm implemented with the help of IBM Research center. However, it is not accessible anymore and already a part of some patents especially in MPEG-4 part 25 (Jovanova et al. 2008). Thanks to two spanning trees Topological Surgery encodes a triangular mesh with about 2.48 to 7 bpv. Hand-and-Glove algorithm (Diaz-Gutierrez et al. 2005) again using two types of spanning trees, encodes genus-0 triangle mesh with 4 bpv guaranteed cost. The trees are encoded with 2 bpv, and one additional bit per triangle allows reconstruction of the triangle strip. (Li and Kuo 1998) encodes connectivity of the triangle mesh with its dual graph. Experimental results from other papers have reported that connectivity compression rates are average of 1.5 bits per triangle not vertex.

(Bajaj et al. 1999) have proposed an alternative representation based on layered decomposition. Mesh connectivity compression performance has been reported around 1.5 to 6 bpv. This method can process nonmanifold meshes too.

The pioneering algorithm of valence-driven approach has been proposed by (Touma and Gotsman 1998). The connectivity is encoded by the valence of the inserted vertices, at around six generally. An entropy coder can efficiently compress these valence values with the performance of 2.4 bpv. A regular mesh can be encoded with almost 0 bpv theoretically. (Alliez and Desbrun 2001a) later proposed some modifications on Touma and Gotsman method to further reduce the compression rates. Valence-based approaches have been claimed as the optimum method by Alliez & Desbrun. The Freelence coder (Kälberer et al. 2005) has a slightly different approach that encodes the number of unconquered edges adjacent to the processed vertex. TFAN (Mamou et al. 2009) has been proposed as an extended valence approach which can compress nonmanifold and non-oriented meshes. Predefined 10 fan configurations have been looked for within mesh data, than its configuration number and valence encoded accordingly. Performance values are stated as 0.2 to 2.7 bpv for connectivity only.

Triangle traversal methods have started with two similar algorithms: EdgeBreaker (Rossignac 1999) and Cut-Border-Machine (Gumhold and Straßer 1998) without knowing each other. Both methods follow the traversing approach of extending the border formed by an initial triangle by iteratively traversing adjacent triangles. Cut-Border-Machine can compress manifold triangles with 4 bpv approximately. Experimental results have been stated from other papers from 3.22 to 8.94 bpv. This method doesn't have a tight upper bound because of the offset it encodes. On the other hand, EdgeBreaker guarantees a performance rate of 4 bpv. Experimental results have been stated from 1.8 to 2.4 bpv. Some later improvements guaranteed the worst-case scenario to 3.67 bpv first and then 3.55 bpv (Gumhold 1999, 2000, King and Rossignac 1999). With high regularity, even better results occurred like 1.622 bpv (Szymczak et al. 2001).

Angle Analyzer (Lee et al. 2002) encodes the connectivity by a gate-based approach with cooperation between geometry and connectivity, in order to achieve an efficient mesh traversal driven by both criteria adaptively. Angle Analyzer can be classified as geometry-driven encoding methods because of the heavily dependent on traversal algorithm. Performance of this method is around 1.5 bpv on average.

7.2. General-Purpose Compressors

As stated in Figure 6.1 general-purpose compressors have been tested on geometry and connectivity separately. Moreover, results are given separately too.

7.2.1. Geometry Information

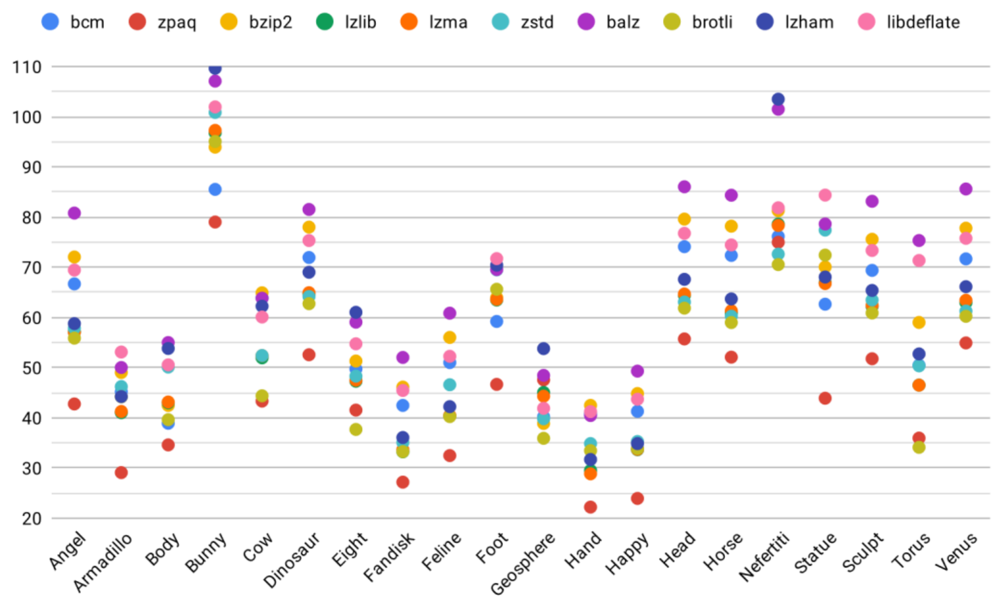


Figure 7.1 Bit per vertex performance representation of selected general-purpose compressors on geometry information only (lower is better)

Geometry	bcm	zpaq	bzip2	lzlib	lzma	zstd	balz	brotli	lzham	libdeflate
Total Rank	106	189	69	145	136	119	33	164	82	57

Figure 7.2 Total ranking of each method for geometry information of all models

Most of the 3D mesh compression algorithms use some prediction and quantization methods. In our work, we do not apply quantization since working with lossless methods. Applying general-purpose compressors directly was a bit risky. However, the results are far better than expected. Total rankings are given in Figure 7.1 which shows that **zpaq** is the best compressor for geometry information without quantization followed by **brotli**. Worst performance values as a bpv have come out from the **balz**. The average bpv values are 44.6 for zpaq, 52 for brotli and 70 for balz.

7.2.2. Connectivity Information

Most of the 3D mesh compression methods are heavily connectivity-based compressors. Connectivity is not stored in explicitly, therefore, a transform or compression needed before applying general-purpose compressors on connectivity information. We have selected the output of connectivity encoder EdgeBreaker, Alliez & Desbrun, and Face Fixer. Figure 7.3 is the graph of the result of EdgeBreaker connectivity tested with general-purpose compressors. All results are available in the Appendix.

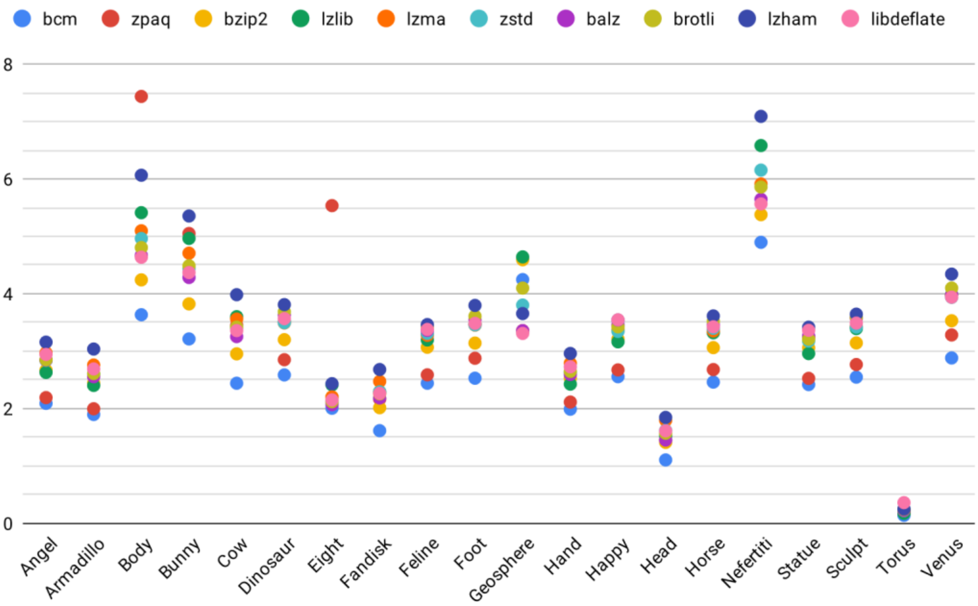


Figure 7.3 Bit per vertex performance representation of selected general-purpose compressors on EdgeBreaker connectivity information only

EdgeBreaker	bcm	zpaq	bzip2	lzlib	lzma	zstd	balz	brotli	lzham	libdeflate
Total Rank	194	131,5	152,5	108	88	110,5	110	81,5	31	93

Alliez Desbrun	bcm	zpaq	bzip2	lzlib	lzma	zstd	balz	brotli	lzham	libdeflate
Total Rank	187,5	118	154,5	93,5	111,5	96,5	82	107	45,5	104

FF labels	bcm	zpaq	bzip2	lzlib	lzma	zstd	balz	brotli	lzham	libdeflate
Total Rank	199,5	135	150	127	57,5	75,5	118,5	120	32	85

Figure 7.4 Total ranking of each method for connectivity information of all models

Total rankings are given in Figure 7.4. According to experimental results, **bcm** is the best general-purpose compressor for connectivity data after 3D mesh compression methods. The second-best compressor is **bzip2**. This time **zpaq** is the third among ten compressors. Worst performance values as a bpv has come out from the **lzham**. The average bpv values are 2.48 for bcm, 2.99 for bzip2, 3.60 for lzham. Nefertiti and Geosphere model somehow broke the zpaq heavy context modeling and results are not even close the worst algorithm. Moreover, those two models have been excluded while visualizing the graph of the result in Figure 7.3.

Total Model	bcm	zpaq	bzip2	lzlib	lzma	zstd	balz	brotli	lzham	libdeflate
Total Rank	110	182	70	147,5	134,5	120	33	164	81	58

Figure 7.5 Total ranking of the total models with EdgeBreaker + CBCM method

The total rankings are also calculated for total models which are connectivity and geometry combined models. Ranking results show that **zpaq** is ahead of others like in geometry compression. However, this time the second best compressor is Google's **brotli** algorithm way better than bcm which is best connectivity compressor.

7.3. Total Compression Results

EdgeBreaker and Alliez & Desbrun data compressed with our CBCM and BCCM approach added to 8 other methods selected available 3D mesh compression methods. As BCCM name stands for best compressors combined method its compression rates are equal or better than CBCM approach. Therefore Figure 7.5 only includes BCCM method for EdgeBreaker and Alliez & Desbrun.

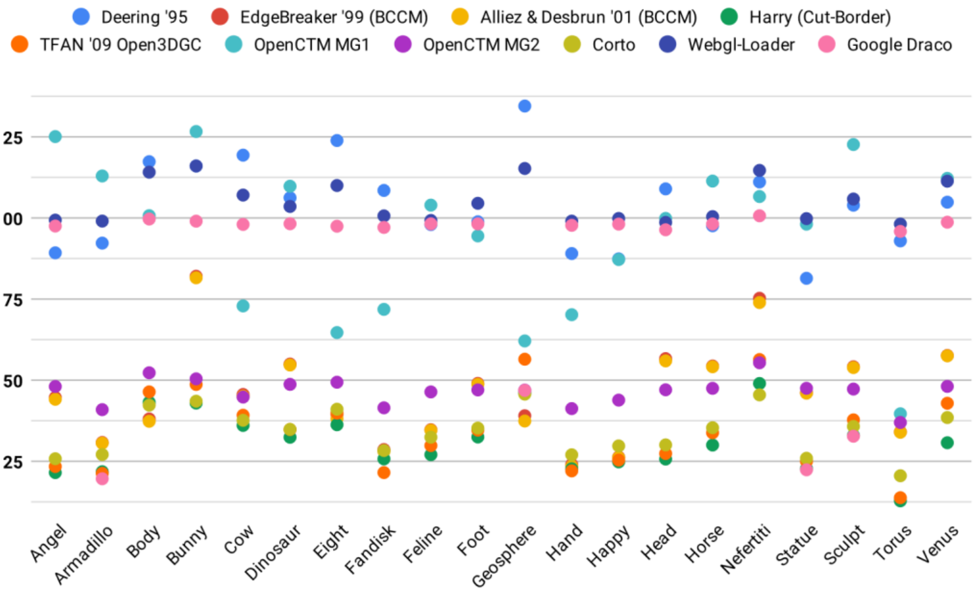


Figure 7.6 Compression performances (bpv) of collected and proposed methods

Deering	EdgeBreaker		Alliez & Desbrun		Harry	TFAN	OpenCTM	OpenCTM	Corto	Webgl-	Google
	w/ BCCM and	w/ CBCM	w/ BCCM and	w/ CBCM	(CBM)	Open3DGC	MG1	MG2		Loader	Draco
47,5	147	110	179	141	225	202	51	136	190	36,5	95

Figure 7.7 Total ranking of each collected methods for all models

Total rankings are given in Figure 7.6. Average ranking method is used. As a result, Alliez & Desbrun with our BCCM approach has the 4th best performance rate among all. If we extracted the Corto and Harry which are variant of progressive mesh compression methods but can be used as a single-rate mesh compressor, Alliez & Desbrun with our BCCM approach has the 2nd best performance rate. Worst performance value as a bpv has come out from the WebGL-Loader followed by Deering.

The average performance (bpv) values are 31.01 for Harry (CBM), 45.42 for Alliez & Desbrun, and 105.01 for WebGL-Loader.

8. CONCLUSION AND FUTURE WORK

General-purpose data compression has come a long way. Current active compression developments groundbreakingly continue. Developments are supported by technology giants and even actively used by them. 3D mesh compression, on the other hand, evolved towards the needs of technology from single-rate mesh compression to progressive and even sequence mesh compression methods. However, single-rate mesh compression, the starting point and main compression branch of 3D mesh, not updated with data compression's groundbreaking ideas.

The easiest way to combine groundbreaking ideas with currently available mesh compression method is to last stage of 3D mesh compression which is general-purpose data compression. This thesis shows that, applying current best general-purpose data compressors to transformed or compressed data of 3D mesh, succeeded in combining groundbreaking ideas with mesh compression methods.

This thesis has been limited with only single-rate triangular mesh compression methods and selected ten general-purpose compressors. Other than triangular there are polygonal mesh compression methods too. Other 3D mesh compression methods generally apply quantization and then use various prediction method to store only prediction errors. In this thesis geometry information left alone without any quantization or prediction at all. Not quantization but prediction methods can be applied before testing the general-purpose compressors since data type is important for the compressors results may change enormously in a better or worse way. It needs to be tested.

REFERENCES

Alliez, P., and M. Desbrun. 2001a. Valence-Driven Connectivity Encoding for 3D Meshes. Computer Graphics Forum 20:480–489.

Alliez, P., and M. Desbrun. 2001b. Valence-Driven Connectivity Encoding for 3D Meshes. http://www.geometry.caltech.edu/SingleRateEncoder/.

Alliez, P., and C. Gotsman. 2005. Recent Advances in Compression of 3D Meshes. Advances in Multiresolution for Geometric Modelling:3–26.

Alliez, P., G. Ucelli, C. Gotsman, and M. Attene. 2008. Recent advances in remeshing of surfaces. Page Mathematics and Visualization.

Bajaj, C. L., V. Pascucci, and G. Zhuang. 1998. Compression and coding of large cad models.

Bajaj, C. L., V. Pascucci, G. Zhuang, and P. Work. 1999. Single Resolution Compression of Arbitrary Triangular Meshes. Computational Geometry: Theory and Applications:1–10.

Bayazıt, U., O. Orcay, U. Konur, and F. S. Gurgen. 2007. Predictive Vector Quantization of 3-D Polygonal Mesh Geometry. Design:1–4.

Botsch, M., M. Pauly, C. Rossl, S. Bischoff, and L. Kobbelt. 2006. Geometric Modeling Based on Triangle Meshes. Page ACM SIGGRAPH 2006 Courses. ACM, New York, NY, USA.

Bouzidi, H. 2013. TurboBench. https://github.com/powturbo/TurboBench.

Brettle, J., and F. Galligan. 2017. Draco – opensource.google.com. https://opensource.google.com/projects/draco.

Buelow, M. Von, S. Guthe, and M. Goesele. 2017. Compression of Non-Manifold Polygonal Meshes Revisited. Page Eurographics Proceedings.

Castelli Aleardi, L., O. Devillers, and G. Schaeffer. 2008. Succinct representations of planar maps. Theoretical Computer Science 408:174–187.

Chou, P. H., and T. H. Meng. 2002. Vertex data compression through vector quantization. IEEE Transactions on Visualization and Computer Graphics 8:373–382.

Chow, M. M. 1997. Optimized geometry compression for real-time rendering. Visualization ’97., Proceedings:347–354.

Chun, W. 2011. webgl-loader. https://code.google.com/archive/p/webgl-loader/.

Cignoni, P. . C., F. . Ganovelli, E. . D. Gobbetti, E. . Martopn, F. . Ponchio, and R. . Scopigno. 2004. Adaptive TetraPuzzles: Efficient out-of-core construction and visualization of gigantic multiresolution polygonal models. ACM Trans. Graph.

Cignoni, P., M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. 2008. MeshLab: an Open-Source Mesh Processing Tool. Page *in* V. Scarano, R. De Chiara, and U. Erra, editors. Eurographics Italian Chapter Conference. The Eurographics Association.

Cignoni, P., F. Ganovelli, and F. Ponchio. 2005. Visualization and Computer Graphics Lib. https://sourceforge.net/projects/vcg/.

Collet, Y. 2013. Finite State Entropy library. https://github.com/Cyan4973/FiniteStateEntropy.

Collet, Y. 2015. Zstandard, zstd. http://www.zstd.net.

Deering, M. 1995a. Geometry compression. Proceedings of the 22nd annual conference on …:13–20.

Deering, M. 1995b. 3D Geometry Compression. https://docs.oracle.com/cd/E17802_01/j2se/javase/technologies/desktop/java3d/forDevelopers/j3dguide/AppendixCompress.doc.html.

Desbrun, M. 2004. The Applied Geometry Lab at Caltech. http://www.geometry.caltech.edu/.

Diaz-Gutierrez, P., M. Gopi, and R. Pajarola. 2005. Hierarchyless Simplification, Stripification and Compression of Triangulated Two-Manifolds. Computer 24.

Diaz, A. D. 2009. Lzlib. https://www.nongnu.org/lzip/lzlib.html.

Duda, J. 2013. Asymmetric numeral systems: entropy coding combining speed of Huffman coding with compression rate of arithmetic coding. arXiv preprint arXiv:1311.2540.

De Floriani, L., and A. Hui. 2003. A Scalable Data Structure for Three-dimensional Non-manifold Objects. Pages 72–82 Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland.

Geelnard, M. 2009. OpenCTM - Compression of 3D triangle meshes. http://openctm.sourceforge.net/.

Geldreich, R. 2009. Lzham. https://github.com/richgel999/lzham_codec.

Google. 2015. Brotli. https://github.com/google/brotli.

Gumhold, S. 1999. Improved Cut-Border Machine for Triangle Mesh Compression. Report.

Gumhold, S. 2000. New Bounds on The Encoding of Planar Triangulations.

Gumhold, S., and W. Straßer. 1998. Real Time Compression of Triangle Mesh Connectivity. SIGGRAPH 98 Proceedings of the 25th annual conference on Computer graphics and interactive techniques 32:133–140.

Gurung, T., D. Laney, P. Lindstrom, and J. Rossignac. 2011a. SQuad: Compact representation for triangle meshes. Computer Graphics Forum 30:355–364.

Gurung, T., M. Luffel, P. Lindstrom, and J. Rossignac. 2011b. LR: Compact Connectivity Representation for Triangle Meshes. ACM SIGGRAPH 2011 papers on - SIGGRAPH '11 1:1.

Gurung, T., M. Luffel, P. Lindstrom, and J. Rossignac. 2013. Zipper: A compact connectivity data structure for triangle meshes. CAD Computer Aided Design 45:262–269.

Gurung, T., and J. Rossignac. 2010. SOT: Compact representation for triangle and tetrahedral meshes. Georgia Institute of Technology GT-IC-10-01:1–10.

Homeomorphic surfaces. (n.d.). . https://www.open.edu/openlearn/science-maths-technology/mathematics-statistics/surfaces/content-section-2.4#.

Isenburg, M. 2000. Triangle Fixer: Edge-based connectivity compression. In 16th EuropeanWorkshop on Comp.Geom. 2:18–23.

Isenburg, M. 2002. Compressing polygon mesh connectivity with degree duality prediction. Graphics Interface:161–170.

Isenburg, M., and P. Alliez. 2002. Compressing polygon mesh geometry with parallelogram prediction. IEEE Visualization, 2002. VIS 2002.:141–146.

Isenburg, M., P. Alliez, and J. Snoeyink. 2002. Benchmark Coding for Polygon Mesh Compression and Triangle Mesh Compression. http://www.cs.unc.edu/~isenburg/pmc/.

Isenburg, M., and P. Lindstrom. 2005. Streaming meshes. Proceedings of the IEEE Visualization Conference:30.

Isenburg, M., and J. Snoeyink. 2000. Face Fixer : Compressing Polygon Meshes with Properties. Proceedings of the 27th annual conference on Computer graphics and interactive techniques:263–270.

Isenburg, M., and J. Snoeyink. 2001. Spirale Reversi: Reverse decoding of the Edgebreaker encoding. Computational Geometry: Theory and Applications 20:39–52.

Jovanova, B., M. Preda, and F. Preteux. 2008. MPEG-4 part 25: A generic model for 3D graphics compression. 2008 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video, 3DTV-CON 2008 Proceedings:101–104.

Kälberer, F., K. Polthier, U. Reitebuch, and M. Wardetzky. 2005. FreeLence - Coding with free valences. Computer Graphics Forum 24:469–478.

Kallmann, M., and D. Thalmann. 2001. Star-Vertices: A Compact Representation for Planar Meshes with Adjacency Information. Journal of Graphics Tools 6:7–18.

Kettner, L. 1999. Using generic programming for designing a data structure for polyhedral surfaces. Computational Geometry: Theory and Applications 13:65–90.

Khodakovsky, A., P. Alliez, M. Desbrun, and P. Schröder. 2002. Near-optimal connectivity encoding of 2-manifold polygon meshes. Graphical Models 64:147–168.

King, D., and J. Rossignac. 1999. Guaranteed 3.67 v bit encoding of planar triangle graphs. Canadian Conference on Computational Geometry:95–98.

Lee, E. S., and H. S. Ko. 2000. Vertex data compression for triangular meshes. Proceedings - Pacific Conference on Computer Graphics and Applications 2000–Janua:225–234.

Lee, H., P. Alliez, and M. Desbrun. 2002. Angle-Analyzer: A triangle-quad mesh codec. Computer Graphics Forum 21:383–392.

Lee, H., and S. Park. 2005. Adaptive Vertex Chasing for the Lossless Geometry Coding of 3D Meshes. Advances in Multimedia Information Processing - PCM 2005 3767:108–119.

Lewiner, T., M. Craizer, H. Lopes, S. Pesco, L. Velho, and E. Medeiros. 2006. GEncode: Geometry-driven compression for General Meshes. Computer Graphics Forum 25:685–695.

Li, J., and C. J. Kuo. 1998. A Dual Graph Approach to 3D Triangular Mesh Compression:1–4.

Lu, Z. M., and Z. Li. 2008. Dynamically restricted codebook-based vector quantisation scheme for mesh geometry compression. Signal, Image and Video Processing 2:251–260.

Luffel, M., T. Gurung, P. Lindstrom, and J. Rossignac. 2014. Grouper: A compact, streamable triangle mesh data structure. IEEE Transactions on Visualization and Computer Graphics 20:84–98.

Maglo, A., G. Lavoué, F. Dupont, and C. Hudelot. 2015. 3D Mesh Compression: Survey, Comparisons, and Emerging Trends. ACM Computing Surveys.

Mahoney, M. 2009. ZPAQ Incremental Journaling Backup Utility and Archiver. http://mattmahoney.net/dc/zpaq.html.

Mamou, K. 2009. Open 3D Graphics Compression. https://github.com/KhronosGroup/glTF/wiki/Open-3D-Graphics-Compression.

Mamou, K., T. Zaharia, and F. Prêteux. 2009. TFAN: A low complexity 3D mesh compression algorithm. Page Computer Animation and Virtual Worlds.

Meng, S., A. Wang, and S. Li. 2010. Compression of 3D triangle meshes based on predictive vector quantization. ISSCAA2010 - 3rd International Symposium on Systems and Control in Aeronautics and Astronautics:1403–1406.

Muravyov, I. 2008. BCM. https://github.com/encode84/bcm.

Muravyov, I. 2016. Balz. https://sourceforge.net/projects/balz/.

Oral, M., and A. A. Elmas. 2017. A Brief History of 3D Mesh Compression. Pages 136–140 2nd International Mediterranean Science and Engineering Congress (IMSEC 2017).

Osfield, R. 2001. OpenSceneGraph. http://www.openscenegraph.org/.

Peng, J., C. S. Kim, and C. C. J. Kuo. 2005. Technologies for 3D mesh compression: A survey. Journal of Visual Communication and Image Representation 16:688–733.

Ponchio, F. 2015. Corto. http://vcg.isti.cnr.it/corto/index.html#overview.

Ponchio, F., and M. Dellepiane. 2015. Fast decompression for web-based view-dependent 3D rendering. Proceedings of the 20th International Conference on 3D Web Technology - Web3D ’15:199–207.

Preda, M. 2008. Graphics Codec - MPEG-4. http://www.mymultimediaworld.com/software/opensource/gc/.

Rchoetzlein. 2009. Elements of polygonal mesh modeling. https://en.wikipedia.org/wiki/Polygon_mesh#/media/File:Mesh_overview.svg .

Rossignac, J. 1999. Edgebreaker: Connectivity compression for triangle meshes. IEEE Transactions on Visualization and Computer Graphics 5:47–61.

Rossignac, J. 2005. 3D mesh compression. Visualization Handbook:359–379.

Rossignac, J., and A. Szymczak. 1999. Wrap & Zip decompression of the connectivity of triangle meshes compressed with Edgebreaker 14:119–135.

Schindler, M. 1998. A fast renormalisation for arithmetic coding. Page 572 Proceedings DCC ’98 Data Compression Conference (Cat. No.98TB100225).

Seward, J. 1996. Bzip2. http://www.bzip.org/.

Sim, J. Y., C. S. Kim, and S. U. Lee. 2003. An efficient 3D mesh compression technique based on triangle fan structure. Signal Processing: Image Communication 18:17–32.

Storer, J. A., and T. G. Szymanski. 1982. Data compression via textual substitution. J. ACM 29:928–951.

Szymczak, A., D. King, and J. Rossignac. 2001. An Edgebreaker-based efficient compression scheme for regular meshes. Computational Geometry: Theory and Applications 20:53–68.

Taubin, G., W. P. Horn, F. Lazarus, and J. Rossignac. 1998. Geometry coding and VRML. Proceedings of the IEEE 86:1228–1243.

Taubin, G., and J. Rossignac. 1998. Geometric compression through topological surgery. ACM Transactions on Graphics 17:84–115.

Taubin, G., and J. Rossignac. 1999. 3D geometry compression. Course Notes 21:18–24.

Touma, C., and C. Gotsman. 1998. Triangle mesh compression. Graphics Interface.

Tunstall, B. P. 1967. Synthesis of noiseless compression codes. Georgia Institute of Technology.

Turán, G. 1984. On the succinct representation of graphs. Discrete Applied Mathematics 8:289–294.

Turk, G., and M. Levoy. 1996. The Stanford 3D Scanning Repository. http://graphics.stanford.edu/data/3Dscanrep/.

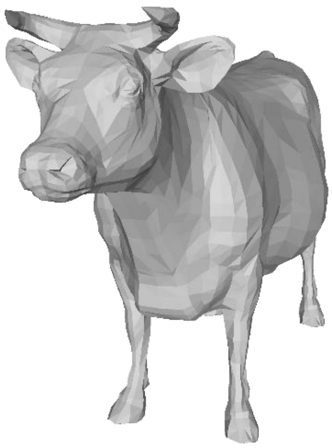
Turk, G., and B. Mullins. 2000. Large Geometric Models Archive. https://www.cc.gatech.edu/projects/large_models/.

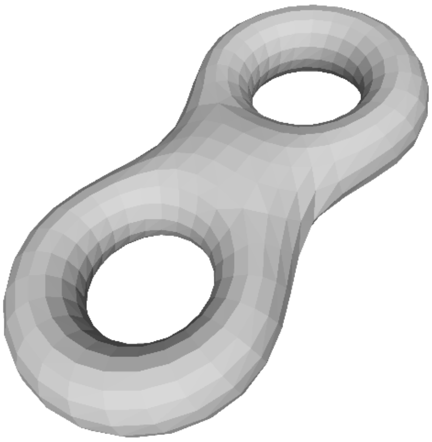
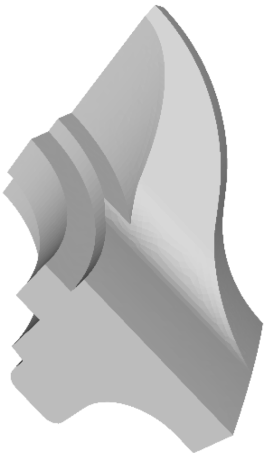
CURRICULUM VITAE

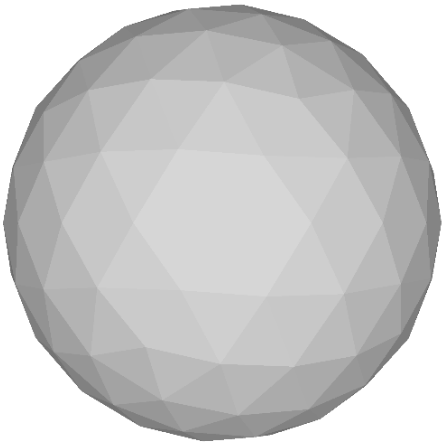
Ammar Abbas ELMAS was born in Konya, in 1990. He completed the elementary school education at İzmir, Turkey. He graduated to the Maltepe Military High School in 2008 and joined the Turkish Military Academy. He left the Turkish Military Academy in 2009. He graduated from the Department of Computer Engineering, KTO Karatay University, Konya, in 2014. He started working as an R&D Engineer and then joined the academy at Çukurova University in 2016 as a Research Assistant.

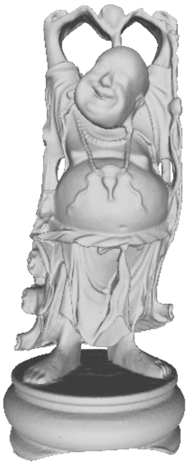
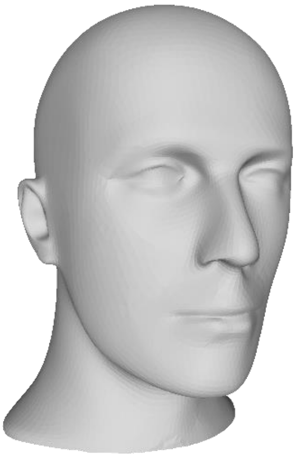
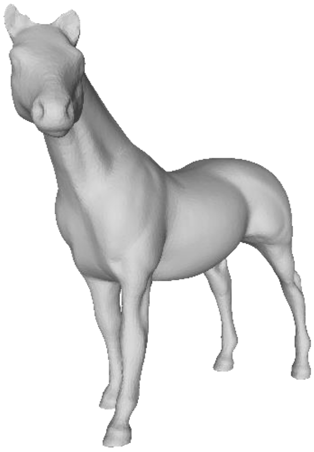
APPENDIX

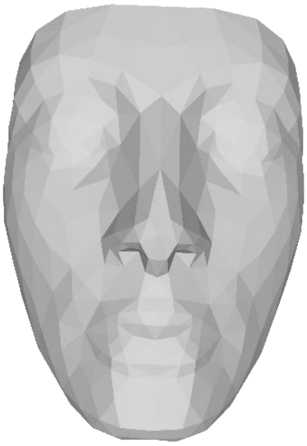
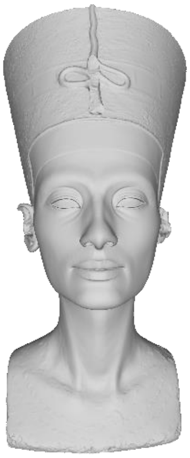
	Mesh info: Angel Vertices: 237018 Faces: 474048 Manifold: YES Edges: 711072 Degenerated faces: 6 Holes: 0 Border edges: 0 Connected components: 1 Genus: 4 Type of Mesh: IRREGULAR Orientable Mesh: YES Oriented Mesh: YES Duplicated vertices: 3
	Mesh info: Armadillo Vertices: 172974 Faces: 345944 Manifold: YES Edges: 518916 Holes: 0 Border edges: 0 Connected components: 1 Genus: 0 Type of Mesh: IRREGULAR Orientable Mesh: YES Oriented Mesh: YES Duplicated vertices: NO
	Mesh info: Body Vertices: 711 Faces: 1396 Manifold: YES Edges: 2082 Holes: 0 Border edges: 24 Connected components: 1 Genus: 0 Type of Mesh: IRREGULAR Orientable Mesh: YES Oriented Mesh: YES Duplicated vertices: NO Self-Intersection: NO

	Mesh info: Bunny Vertices: 1494 Faces: 2915 Manifold: YES Edges: 4333 Holes: 0 Border edges: 79 Connected components: 1 Genus: 2 Type of Mesh: IRREGULAR Orientable Mesh: YES Oriented Mesh: YES Duplicated vertices: NO Self-Intersection: YES
	Mesh info: Cow Vertices: 2904 Faces: 5804 Manifold: YES Edges: 8706 Holes: 0 Border edges: 0 Connected Components: 1 Genus: 0 Type of Mesh: IRREGULAR Orientable Mesh: YES Oriented Mesh: YES Duplicated vertices: 1 Self-Intersection: YES
	Mesh info: Dinosaur Vertices: 14070 Faces: 28136 Manifold: YES Edges: 42204 Holes: 0 Border edges: 0 Connected Components: 1 Genus: 0 Type of Mesh: IRREGULAR Orientable Mesh: YES Oriented Mesh: YES Duplicated vertices: 4 Self-Intersection: YES

	Mesh info: Eight Vertices: 766 Faces: 1536 Manifold: YES Edges: 2304 Holes: 0 Border edges: 0 Connected Components: 1 Genus: 2 Type of Mesh: IRREGULAR Orientable Mesh: YES Oriented Mesh: YES Duplicated vertices: NO Self-Intersection: NO
	Mesh info: Fandisk Vertices: 6475 Faces: 12946 Manifold: YES Edges: 19419 Holes: 0 Border edges: 0 Connected Components: 1 Genus: 0 Type of Mesh: IRREGULAR Orientable Mesh: YES Oriented Mesh: YES Duplicated vertices: NO Self-Intersection: YES
	Mesh info: Feline Vertices: 49864 Faces: 99732 Manifold: YES Edges: 149598 Holes: 0 Border edges: 0 Connected Components: 1 Genus: 2 Type of Mesh: IRREGULAR Orientable Mesh: YES Oriented Mesh: YES Duplicated vertices: NO Self-Intersection: YES

	Mesh info: Foot Vertices: 10016 Faces: 20028 Manifold: YES Edges: 30042 Holes: 0 Border edges: 0 Connected Components: 1 Genus: 0 Type of Mesh: IRREGULAR Orientable Mesh: YES Oriented Mesh: YES Duplicated vertices: NO Self-Intersection: NO
	Mesh info: Geosphere Vertices: 162 Faces: 320 Manifold: YES Edges: 480 Holes: 0 Border edges: 0 Connected Components: 1 Genus: 0 Type of Mesh: SEMIREGULAR Orientable Mesh: YES Oriented Mesh: YES Duplicated vertices: NO Self-Intersection: NO
	Mesh info: Hand Vertices: 327323 Faces: 654666 Manifold: YES Edges: 981999 Holes: 0 Border edges: 0 Connected Components: 1 Genus: 6 Type of Mesh: IRREGULAR Orientable Mesh: YES Oriented Mesh: YES Duplicated vertices: NO

	Mesh info: Happy Vertices: 543652 Faces: 1087716 Manifold: YES Edges: 1631574 Degenerated faces: 2080 Holes: 0 Border edges: 0 Connected Components: 1 Genus: 104 Type of Mesh: IRREGULAR Orientable Mesh: YES Oriented Mesh: YES Duplicated vertices: 1040
	Mesh info: Head Vertices: 11703 Faces: 23402 Manifold: YES Edges: 35103 Holes: 0 Border edges: 0 Connected Components: 1 Genus: 0 Type of Mesh: IRREGULAR Orientable Mesh: YES Oriented Mesh: YES Duplicated vertices: NO Self-Intersection: YES
	Mesh info: Horse Vertices: 19851 Faces: 39698 Manifold: YES Edges: 59547 Holes: 0 Border edges: 0 Connected Components: 1 Genus: 0 Type of Mesh: IRREGULAR Orientable Mesh: YES Oriented Mesh: YES Duplicated vertices: NO Self-Intersection: YES

	Mesh info: Nefertiti Vertices: 299 Faces: 562 Manifold: YES Edges: 826 Holes: 0 Border edges: 34 Connected Components: 1 Genus: 0 Type of Mesh: IRREGULAR Orientable Mesh: YES Oriented Mesh: YES Duplicated vertices: NO Self-Intersection: NO
	Mesh info: Statue Vertices: 1009118 Faces: 2018232 Manifold: YES Edges: 3027348 Holes: 0 Border edges: 0 Connected Components: 1 Genus: 0 Type of Mesh: IRREGULAR Orientable Mesh: YES Oriented Mesh: YES Duplicated vertices: NO
	Mesh info: Sculpt Vertices: 21469 Faces: 42934 Manifold: YES Edges: 64401 Degenerated faces: 1310 Holes: 0 Border edges: 0 Connected Components: 1 Genus: 0 Type of Mesh: IRREGULAR Orientable Mesh: YES Oriented Mesh: YES Duplicated vertices: 655 Self-Intersection: YES

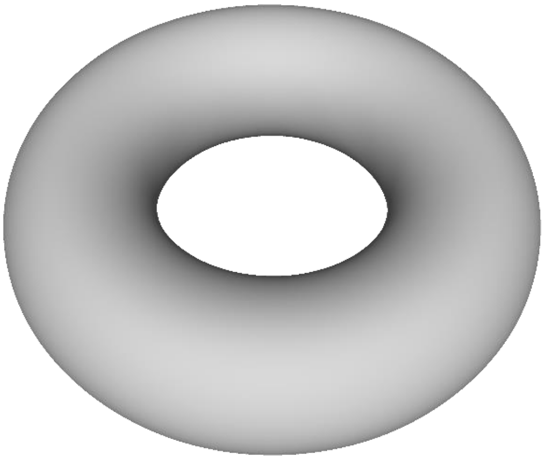
	Mesh info: Torus Vertices: 36450 Faces: 72900 Manifold: YES Edges: 109350 Holes: 0 Border edges: 0 Connected Components: 1 Genus: 1 Type of Mesh: REGULAR Orientable Mesh: YES Oriented Mesh: YES Duplicated vertices: NO Self-Intersection: NO
	Mesh info: Venus Vertices: 8268 Faces: 16532 Manifold: YES Edges: 24798 Holes: 0 Border edges: 0 Connected Components: 1 Genus: 0 Type of Mesh: IRREGULAR Orientable Mesh: YES Oriented Mesh: YES Duplicated vertices: NO Self-Intersection: YES

Table 0.1 Compression Ratio (Uncompressed / Compressed)

Compression Ratio	Deering	EdgeBreaker w/ BCCM	EdgeBreaker and w/ CBCM	Alliez & Desbrun w/ BCCM	Alliez & Desbrun and w/ CBCM	Harry (CBM)	TFAN Open3DGC	OpenCTM MG1	OpenCTM MG2	Corto	Webgl-Loader	Google Draco
Angel	7	14	14	14	14	29	27	5	13	24	7	7
Armadillo	7	21	21	21	21	29	30	6	16	23	7	32
Body	5	13	12	13	12	12	11	5	10	12	5	5
Bunny	5	7	7	7	7	13	11	5	11	12	5	6
Cow	5	12	12	12	12	15	14	8	12	14	5	6
Dinosaur	6	10	10	11	11	17	16	6	12	16	6	6
Eight	4	13	11	13	11	14	13	8	10	12	5	5
Fandisk	5	19	18	19	19	21	25	8	13	19	6	6
Feline	6	17	17	17	17	22	20	6	13	18	6	6
Foot	6	12	12	12	12	18	17	7	13	17	6	6
Geosphere	4	12	7	13	8	11	9	8	10	11	5	10
Hand	7	26	26	26	26	27	28	9	15	23	7	7
Happy	8	24	24	24	24	25	25	8	15	21	7	7
Head	5	10	10	10	10	21	20	6	12	18	6	6
Horse	6	11	11	11	11	19	17	6	12	16	6	6
Nefertiti	5	7	6	7	6	10	9	5	9	11	5	5
Statue	9	15	15	15	15	29	27	7	14	26	7	30
Sculpt	6	11	11	11	11	18	15	5	12	16	6	18
Torus	7	17	16	17	16	45	42	15	16	28	6	6
Venus	6	10	10	10	10	18	13	5	12	14	5	6

Table 0.2 Storage cost in percentage

Storage Cost %	Deering	EdgeBreaker w/ BCCM	EdgeBreaker and w/ CBCM	Alliez & Desbrun w/ BCCM	Alliez & Desbrun and w/ CBCM	Harry (CBM)	TFAN Open3DGC	OpenCTM MG1	OpenCTM MG2	Corto	Webgl-Loader	Google Draco
Angel	14,571	7,3048	7,3211	7,214	7,214	3,5315	3,8374	20,411	7,8576	4,2268	16,2155	15,9176
Armadillo	14,8708	4,9814	4,9974	4,9504	4,9576	3,5303	3,448	18,2017	6,609	4,3844	15,9637	3,1822
Body	24,3782	7,9262	8,7148	7,7768	8,6192	8,9925	9,6575	20,9249	10,8778	8,8082	23,7062	20,7103
Bunny	22,3327	15,7987	16,1517	15,7092	16,0755	8,2852	9,3883	24,3752	9,7196	8,3943	22,3327	19,0647
Cow	22,9754	8,7936	8,9874	8,6846	8,8715	6,962	7,5649	14,042	8,6396	7,2637	20,6106	18,8721
Dinosaur	19,3113	10,006	10,0541	9,9537	9,9895	5,9099	6,3435	19,956	8,8709	6,352	18,8238	17,8566
Eight	25,4352	8,1337	9,6474	7,8532	9,247	7,47	8,2065	13,2957	10,1569	8,4506	22,5919	20,0184
Fandisk	20,5877	5,4519	5,5558	5,3409	5,4282	4,9047	4,1069	13,6403	7,8894	5,3952	19,0997	18,4276
Feline	17,1253	6,0898	6,1153	6,0581	6,07	4,7471	5,2345	18,1714	8,1223	5,6842	17,3479	17,1639
Foot	17,2297	8,557	8,6175	8,5074	8,5588	5,6811	6,0496	16,471	8,2083	6,1589	18,22	17,1169
Geosphere	28,8372	8,3932	14,7886	8,0444	13,6892	9,9049	12,1247	13,3298	10,0846	9,8203	24,7146	10,0634
Hand	14,6817	3,9787	3,9991	3,8911	3,9461	3,7595	3,6542	11,5788	6,8138	4,4698	16,3259	16,1147
Happy	14,0305	4,2415	4,2604	4,2124	4,2129	4,002	4,0892	13,9981	7,0533	4,7894	16,0267	15,7479
Head	20,1338	10,4778	10,5417	10,3515	10,389	4,7682	5,0828	18,4543	8,7078	5,5734	18,2254	17,8067
Horse	17,444	9,7283	9,7672	9,6889	9,7168	5,382	6,0572	19,9024	8,5024	6,3353	17,9464	17,5409
Nefertiti	23,724	16,0764	18,9507	15,797	18,58	10,4762	12,0502	22,7659	11,8392	9,7291	24,4825	21,4999
Statue	12,3531	7,0114	7,0277	6,9906	6,9937	3,4697	3,8073	14,8819	7,2182	3,951	15,1398	3,4124
Sculpt	18,4383	9,6155	9,6544	9,5736	9,5973	5,8692	6,7172	21,7518	8,3986	6,3601	18,7752	5,8247
Torus	16,2416	5,9724	6,304	5,9498	6,2813	2,2617	2,43	6,9436	6,4756	3,6115	17,1455	16,7497
Venus	19,6947	10,8335	10,9086	10,8132	10,8787	5,7894	8,0715	21,0699	9,0468	7,2462	20,9071	18,537

Table 0.3 Space savings in percentage

Space Savings %	Deering	EdgeBreaker w/ BCCM	and w/ CBCM	Alliez & Desbrun w/ BCCM	and w/ CBCM	Harry (CBM)	TFAN Open3DGC	OpenCTM MG1	OpenCTM MG2	Corto	Webgl-Loader	Google Draco
Angel	85,429	92,6952	92,6789	92,786	92,786	96,4685	96,1626	79,589	92,1424	95,7732	83,7845	84,0824
Armadillo	85,1292	95,0186	95,0026	95,0496	95,0424	96,4697	96,552	81,7983	93,391	95,6156	84,0363	96,8178
Body	75,6218	92,0738	91,2852	92,2232	91,3808	91,0075	90,3425	79,0751	89,1222	91,1918	76,2938	79,2897
Bunny	77,6673	84,2013	83,8483	84,2908	83,9245	91,7148	90,6117	75,6248	90,2804	91,6057	77,6673	80,9353
Cow	77,0246	91,2064	91,0126	91,3154	91,1285	93,038	92,4351	85,958	91,3604	92,7363	79,3894	81,1279
Dinosaur	80,6887	89,994	89,9459	90,0463	90,0105	94,0901	93,6565	80,044	91,1291	93,648	81,1762	82,1434
Eight	74,5648	91,8663	90,3526	92,1468	90,753	92,53	91,7935	86,7043	89,8431	91,5494	77,4081	79,9816
Fandisk	79,4123	94,5481	94,4442	94,6591	94,5718	95,0953	95,8931	86,3597	92,1106	94,6048	80,9003	81,5724
Feline	82,8747	93,9102	93,8847	93,9419	93,93	95,2529	94,7655	81,8286	91,8777	94,3158	82,6521	82,8361
Foot	82,7703	91,443	91,3825	91,4926	91,4412	94,3189	93,9504	83,529	91,7917	93,8411	81,78	82,8831
Geosphere	71,1628	91,6068	85,2114	91,9556	86,3108	90,0951	87,8753	86,6702	89,9154	90,1797	75,2854	89,9366
Hand	85,3183	96,0213	96,0009	96,1089	96,0539	96,2405	96,3458	88,4212	93,1862	95,5302	83,6741	83,8853
Happy	85,9695	95,7585	95,7396	95,7876	95,7871	95,998	95,9108	86,0019	92,9467	95,2106	83,9733	84,2521
Head	79,8662	89,5222	89,4583	89,6485	89,611	95,2318	94,9172	81,5457	91,2922	94,4266	81,7746	82,1933
Horse	82,556	90,2717	90,2328	90,3111	90,2832	94,618	93,9428	80,0976	91,4976	93,6647	82,0536	82,4591
Nefertiti	76,276	83,9236	81,0493	84,203	81,42	89,5238	87,9498	77,2341	88,1608	90,2709	75,5175	78,5001
Statue	87,6469	92,9886	92,9723	93,0094	93,0063	96,5303	96,1927	85,1181	92,7818	96,049	84,8602	96,5876
Sculpt	81,5617	90,3845	90,3456	90,4264	90,4027	94,1308	93,2828	78,2482	91,6014	93,6399	81,2248	94,1753
Torus	83,7584	94,0276	93,696	94,0502	93,7187	97,7383	97,57	93,0564	93,5244	96,3885	82,8545	83,2503
Venus	80,3053	89,1665	89,0914	89,1868	89,1213	94,2106	91,9285	78,9301	90,9532	92,7538	79,0929	81,463

Table 0.4 Total bits per vertex (bpv)

bpv	Deering	EdgeBreaker w/ BCCM	and w/ CBCM	Alliez & Desbrun w/ BCCM	and w/ CBCM	Harry (CBM)	TFAN Open3DGC	OpenCTM MG1	OpenCTM MG2	Corto	Webgl-Loader	Google Draco
Angel	89,4439	44,8405	44,9404	44,2829	44,2831	21,6783	23,556	125,2924	48,2336	25,9459	99,5385	97,7095
Armadillo	92,4417	30,9658	31,0654	30,7732	30,8179	21,9456	21,4342	113,1479	41,0836	27,2549	99,2358	19,7817
Body	117,5584	38,2222	42,0253	37,5021	41,564	43,3643	46,571	100,9058	52,4557	42,4754	114,3179	99,8706
Bunny	116,2195	82,2169	84,0535	81,751	83,6573	43,1165	48,8568	126,8487	50,581	43,6841	116,2195	99,2129
Cow	119,5592	45,7603	46,7686	45,1928	46,1653	36,2287	39,3664	73,0716	44,9587	37,7989	107,2534	98,2066
Dinosaur	106,4256	55,1437	55,4087	54,8554	55,0527	32,5697	34,9595	109,9787	48,8881	35,0061	103,739	98,4091
Eight	124,0731	39,6762	47,0601	38,3081	45,107	36,4386	40,0313	64,8564	49,5457	41,2219	110,2037	97,6501
Fandisk	108,6863	28,7815	29,33	28,1958	28,6567	25,8928	21,6809	72,0099	41,6494	28,4825	100,8309	97,2825
Feline	98,1563	34,9044	35,0505	34,7226	34,7913	27,2084	30,0021	104,1521	46,5541	32,5797	99,4322	98,3772
Foot	99,0224	49,1789	49,5264	48,8938	49,1893	32,6502	34,7684	94,6621	47,1749	35,3962	104,7141	98,3738
Geosphere	134,716	39,2099	69,0864	37,5802	63,9506	46,2716	56,642	62,2716	47,1111	45,8765	115,4568	47,0123
Hand	89,2247	24,1794	24,3037	23,6476	23,9813	22,8478	22,2076	70,3675	41,4093	27,1642	99,2172	97,9338
Happy	87,5816	26,4763	26,5942	26,2948	26,298	24,9817	25,5261	87,3797	44,0285	29,8965	100,0429	98,3022
Head	109,1659	56,8107	57,1573	56,1258	56,3295	25,8532	27,5587	100,0591	47,2139	30,2193	98,8184	96,5482
Horse	97,7909	54,5371	54,7551	54,3159	54,4726	30,1716	33,9566	111,5732	47,6647	35,5158	100,6075	98,3342
Nefertiti	111,3043	75,4247	88,9097	74,1137	87,1706	49,1505	56,5351	106,8094	55,5452	45,6455	114,8629	100,8696
Statue	81,5886	46,3081	46,416	46,1711	46,1917	22,9163	25,146	98,2904	47,6741	26,0949	99,9937	22,5382
Sculpt	104,1338	54,3053	54,5248	54,0687	54,2024	33,1473	37,9364	122,8469	47,4329	35,9197	106,0361	32,8958
Torus	93,1292	34,2459	36,1471	34,116	36,0171	12,9688	13,9334	39,8145	37,1312	20,7085	98,312	96,0426
Venus	105,0643	57,7929	58,1935	57,6846	58,0339	30,8844	43,0585	112,4006	48,2612	38,656	111,5317	98,8882

Table 0.5 Geometry information only compression ratio

Geometry Comp. Rati.	bcm	zpaq	bzip2	lzlib	lzma	zstd	balz	brotli	lzham	libdeflate
Angel	4	6	4	5	5	5	3	5	5	4
Armadillo	6	9	6	7	7	6	6	6	6	5
Body	7	7	6	6	6	5	5	6	5	5
Bunny	4	4	3	3	3	3	3	3	3	3
Cow	4	6	4	5	5	5	4	6	4	4
Dinosaur	4	5	4	4	4	4	3	4	4	4
Eight	5	6	5	5	5	5	4	7	4	5
Fandisk	6	9	6	7	7	7	5	7	7	6
Feline	5	8	5	6	6	5	4	6	6	5
Foot	5	6	5	5	5	4	4	5	4	4
Geosphere	6	5	7	6	6	6	5	7	5	6
Hand	6	11	6	8	8	7	6	7	8	6
Happy	6	10	6	7	7	7	5	7	7	6
Head	4	5	3	4	4	4	3	4	4	4
Horse	4	5	4	4	4	4	3	4	4	4
Nefertiti	4	4	3	4	4	4	3	4	3	3
Statue	5	7	4	4	4	4	4	4	4	4
Sculpt	4	5	4	4	4	4	3	4	4	4
Torus	5	7	4	6	6	5	4	7	5	4
Venus	4	5	4	4	4	4	3	5	4	4

Table 0.6 Geometry information only storage cost in percentage

Geometry Stor. Cost.	bcm	zpaq	bzip2	lzlib	lzma	zstd	balz	brotli	lzham	libdeflate
Angel	28,1888	18,0734	30,4543	24,1593	24,1121	24,4505	34,1547	23,6183	24,8444	29,3553
Armadillo	17,8885	11,5145	19,4154	16,2436	16,3407	18,3053	19,8115	17,4984	17,514	21,0336
Body	16,4792	14,6534	17,9855	18,2096	18,2763	21,227	23,2958	16,7843	22,7953	21,4129
Bunny	33,0881	30,5706	36,3474	37,4498	37,6383	39,0224	41,4405	36,7867	42,4268	39,4513
Cow	27,2461	18,4778	27,6844	22,1546	22,3438	22,3508	27,2262	18,9267	26,5458	25,6245
Dinosaur	30,3589	22,1805	32,9147	27,3002	27,3993	27,0756	34,4007	26,4772	29,1193	31,7852
Eight	21,071	17,5968	21,7393	20,0354	20,1239	20,4647	25,0232	15,9637	25,8553	23,1954
Fandisk	18,304	11,7116	19,8785	14,3177	14,5116	15,1183	22,425	14,3321	15,5465	19,5887
Feline	21,999	14,0031	24,1596	17,4182	17,4521	20,0905	26,2431	17,3551	18,2094	22,5352
Foot	21,4581	16,9109	23,1812	23,0048	23,0555	25,1834	25,2109	23,7834	25,5239	25,9978
Geosphere	17,1044	20,2353	16,5371	19,1637	18,8485	16,9363	20,6136	15,2763	22,883	17,8189
Hand	18,217	9,9065	18,958	13,1745	12,8682	15,5595	18,057	14,9306	14,1419	18,4041
Happy	17,6506	10,2277	19,1657	14,3804	14,4179	15,0782	21,0758	14,4821	14,8994	18,6797
Head	31,4384	23,6363	33,7718	27,3191	27,4345	26,7581	36,5035	26,2395	28,6777	32,5661
Horse	30,7352	22,1205	33,2095	25,8522	26,0466	25,5878	35,8292	25,0496	27,0487	31,6204
Nefertiti	31,8772	31,3844	34,0166	32,8965	32,7845	30,3987	42,4843	29,5251	43,3244	34,263
Statue	23,6071	16,542	26,4029	25,2181	25,1567	29,1723	29,6293	27,2934	25,6454	31,7998
Sculpt	29,128	21,732	31,7282	26,1135	26,2021	26,6486	34,9111	25,5612	27,45	30,794
Torus	21,3958	15,2317	25,0002	19,7085	19,7033	21,3344	31,9194	14,4533	22,3404	30,233
Venus	29,6606	22,7279	32,1945	26,0471	26,2457	25,3538	35,4252	24,9245	27,3671	31,3559

Table 0.7 Geometry information only space savings in percentage

Geometry Spac. Sav.	bcm	zpaq	bzip2	lzlib	lzma	zstd	balz	brotli	lzham	libdeflate
Angel	71,8112	81,9266	69,5457	75,8407	75,8879	75,5495	65,8453	76,3817	75,1556	70,6447
Armadillo	82,1115	88,4855	80,5846	83,7564	83,6593	81,6947	80,1885	82,5016	82,486	78,9664
Body	83,5208	85,3466	82,0145	81,7904	81,7237	78,773	76,7042	83,2157	77,2047	78,5871
Bunny	66,9119	69,4294	63,6526	62,5502	62,3617	60,9776	58,5595	63,2133	57,5732	60,5487
Cow	72,7539	81,5222	72,3156	77,8454	77,6562	77,6492	72,7738	81,0733	73,4542	74,3755
Dinosaur	69,6411	77,8195	67,0853	72,6998	72,6007	72,9244	65,5993	73,5228	70,8807	68,2148
Eight	78,929	82,4032	78,2607	79,9646	79,8761	79,5353	74,9768	84,0363	74,1447	76,8046
Fandisk	81,696	88,2884	80,1215	85,6823	85,4884	84,8817	77,575	85,6679	84,4535	80,4113
Feline	78,001	85,9969	75,8404	82,5818	82,5479	79,9095	73,7569	82,6449	81,7906	77,4648
Foot	78,5419	83,0891	76,8188	76,9952	76,9445	74,8166	74,7891	76,2166	74,4761	74,0022
Geosphere	82,8956	79,7647	83,4629	80,8363	81,1515	83,0637	79,3864	84,7237	77,117	82,1811
Hand	81,783	90,0935	81,042	86,8255	87,1318	84,4405	81,943	85,0694	85,8581	81,5959
Happy	82,3494	89,7723	80,8343	85,6196	85,5821	84,9218	78,9242	85,5179	85,1006	81,3203
Head	68,5616	76,3637	66,2282	72,6809	72,5655	73,2419	63,4965	73,7605	71,3223	67,4339
Horse	69,2648	77,8795	66,7905	74,1478	73,9534	74,4122	64,1708	74,9504	72,9513	68,3796
Nefertiti	68,1228	68,6156	65,9834	67,1035	67,2155	69,6013	57,5157	70,4749	56,6756	65,737
Statue	76,3929	83,458	73,5971	74,7819	74,8433	70,8277	70,3707	72,7066	74,3546	68,2002
Sculpt	70,872	78,268	68,2718	73,8865	73,7979	73,3514	65,0889	74,4388	72,55	69,206
Torus	78,6042	84,7683	74,9998	80,2915	80,2967	78,6656	68,0806	85,5467	77,6596	69,767
Venus	70,3394	77,2721	67,8055	73,9529	73,7543	74,6462	64,5748	75,0755	72,6329	68,6441

Table 0.8 Geometry information only bit per vertex (bpv)

Geometry bpv	bcm	zpaq	bzip2	lzlib	lzma	zstd	balz	brotli	lzham	libdeflate
Angel	66,6758	42,7496	72,0344	57,1447	57,0331	57,8333	80,787	55,865	58,7652	69,435
Armadillo	45,1604	29,069	49,0153	41,0079	41,2531	46,2128	50,0153	44,1758	44,2152	53,1006
Body	38,8973	34,5879	42,4529	42,9817	43,1392	50,1041	54,9873	39,6174	53,8059	50,5429
Bunny	85,51	79,004	93,9331	96,7818	97,2691	100,8461	107,095	95,0683	109,6439	101,9545
Cow	63,876	43,3196	64,9036	51,9394	52,3829	52,3994	63,8292	44,3719	62,2342	60,0744
Dinosaur	71,9369	52,5578	77,9929	64,6891	64,924	64,1569	81,5141	62,7389	68,9996	75,3166
Eight	49,7232	41,5248	51,3003	47,2794	47,4883	48,2924	59,0496	37,671	61,0131	54,7363
Fandisk	42,4587	27,1666	46,1109	33,212	33,6618	35,069	52,0179	33,2454	36,0624	45,4388
Feline	50,9998	32,4632	56,0088	40,3802	40,4588	46,5753	60,8388	40,2341	42,2145	52,2429
Foot	59,1941	46,6502	63,9473	63,4609	63,6006	69,4704	69,5463	65,6086	70,4097	71,7173
Geosphere	40,1975	47,5556	38,8642	45,037	44,2963	39,8025	48,4444	35,9012	53,7778	41,8765
Hand	40,8061	22,1905	42,466	29,5109	28,8248	34,8532	40,4477	33,4445	31,6778	41,2252
Happy	41,2819	23,9209	44,8254	33,6334	33,7209	35,2653	49,2927	33,8712	34,8473	43,6886
Head	74,0938	55,7061	79,5933	64,3855	64,6576	63,0635	86,0313	61,8412	67,5875	76,7516
Horse	72,3567	52,076	78,1817	60,861	61,3188	60,2388	84,3488	58,9717	63,678	74,4406
Nefertiti	76,1472	74,9699	81,2575	78,5819	78,3144	72,6154	101,4849	70,5284	103,4916	81,8462
Statue	62,635	43,8899	70,0531	66,9094	66,7466	77,4008	78,6133	72,4157	68,0432	84,3723
Sculpt	69,3726	51,758	75,5653	62,1931	62,404	63,4675	83,1457	60,8777	65,3761	73,3404
Torus	50,4891	35,9432	58,9946	46,5075	46,4953	50,3443	75,3222	34,1063	52,7181	71,3429
Venus	71,6613	54,9115	77,7833	62,9308	63,4107	61,2559	85,5888	60,2187	66,12	75,7571

Table 0.9 EdgeBreaker CLERS only compression ratio

EdgeBreaker Comp. Rati.	bcm	zpaq	bzip2	lzlib	lzma	zstd	balz	brotli	lzham	libdeflate
Angel	23	22	18	19	17	17	17	17	16	17
Armadillo	26	25	20	20	18	19	19	19	16	18
Body	14	7	12	9	10	10	11	10	8	11
Bunny	15	10	13	10	11	11	12	11	9	12
Cow	20	14	17	14	14	15	15	15	13	15
Dinosaur	19	17	15	14	14	14	14	14	13	14
Eight	24	9	22	20	22	23	24	23	20	23
Fandisk	30	23	24	21	20	21	22	22	18	22
Feline	20	19	16	16	15	15	15	15	14	15
Foot	19	17	16	14	14	14	14	14	13	14
Geosphere	12	3	11	11	15	13	15	12	13	15
Hand	25	23	19	20	18	19	19	19	17	18
Happy	19	18	15	16	15	15	14	15	14	14
Head	44	34	35	32	27	30	34	31	27	30
Horse	20	18	16	15	15	15	14	14	14	15
Nefertiti	10	4	9	8	9	8	9	9	7	9
Statue	20	20	16	17	15	16	15	15	15	15
Sculpt	19	18	16	15	14	15	14	14	14	14
Torus	344	236	262	275	214	217	214	198	192	134
Venus	17	15	14	13	13	13	13	12	12	13

Table 0.10 EdgeBreaker CLERS only storage cost in percentage

EdgeBreaker Stor. Cost %	bcm	zpaq	bzip2	lzlib	lzma	zstd	balz	brotli	lzham	libdeflate
Angel	4,356	4,564	5,5572	5,4771	6,1891	5,9138	5,9239	5,9375	6,5769	6,1314
Armadillo	3,9517	4,1593	5,0929	5,0063	5,7475	5,4154	5,3215	5,4355	6,3258	5,6054
Body	7,5911	15,5347	8,8602	11,3043	10,6463	10,3643	9,7532	10,0353	12,6675	9,6827
Bunny	6,6793	10,4976	7,9483	10,3195	9,7851	9,2285	8,9057	9,3287	11,1321	9,0838
Cow	5,0882	7,19	6,1563	7,5059	7,4197	7,1096	6,7823	7,1269	8,3041	7,0005
Dinosaur	5,3886	5,9408	6,6682	7,3341	7,2761	7,2855	7,5664	7,6789	7,9408	7,4325
Eight	4,1712	11,5142	4,584	5,0185	4,584	4,4102	4,3015	4,4102	5,0619	4,4753
Fandisk	3,3654	4,5087	4,1997	4,7791	5,155	4,7739	4,5525	4,7018	5,585	4,7224
Feline	5,0857	5,3901	6,3915	6,6572	6,8233	6,8935	7,0042	7,1388	7,2147	7,0382
Foot	5,2705	5,9947	6,5457	7,2416	7,1983	7,2066	7,3914	7,5229	7,9141	7,2615
Geosphere	8,977	45,5115	9,7077	9,8121	7,0981	8,0376	7,0981	8,6639	7,7244	6,9937
Hand	4,1435	4,4023	5,2661	5,0539	5,8116	5,4788	5,4047	5,5132	6,1657	5,6813
Happy	5,3217	5,5673	6,6876	6,5869	7,0185	6,9629	7,2021	7,1301	7,3785	7,3836
Head	2,302	3,0242	2,9387	3,1766	3,7407	3,376	3,0242	3,272	3,8461	3,3604
Horse	5,129	5,5833	6,3803	6,9187	6,9859	7,0808	7,2076	7,3386	7,5309	7,1354
Nefertiti	10,2464	29,1713	11,2542	13,7738	12,374	12,8779	11,8141	12,262	14,8376	11,6461
Statue	5,038	5,2628	6,3698	6,1615	6,7491	6,6129	6,8076	6,6856	7,12	7,0056
Sculpt	5,3076	5,7649	6,5491	7,0708	7,1485	7,1236	7,4427	7,4815	7,5933	7,2688
Torus	0,2908	0,4248	0,3827	0,3644	0,4687	0,4614	0,4678	0,5053	0,5231	0,7517
Venus	6,0048	6,8395	7,3557	8,3054	8,2853	8,2046	8,2913	8,5434	9,0475	8,2107

Table 0.11 EdgeBreaker CLERS only space savings in percentage

EdgeBreaker Space Sav. %	bcm	zpaq	bzip2	lzlib	lzma	zstd	balz	brotli	lzham	libdeflate
Angel	95,644	95,436	94,4428	94,5229	93,8109	94,0862	94,0761	94,0625	93,4231	93,8686
Armadillo	96,0483	95,8407	94,9071	94,9937	94,2525	94,5846	94,6785	94,5645	93,6742	94,3946
Body	92,4089	84,4653	91,1398	88,6957	89,3537	89,6357	90,2468	89,9647	87,3325	90,3173
Bunny	93,3207	89,5024	92,0517	89,6805	90,2149	90,7715	91,0943	90,6713	88,8679	90,9162
Cow	94,9118	92,81	93,8437	92,4941	92,5803	92,8904	93,2177	92,8731	91,6959	92,9995
Dinosaur	94,6114	94,0592	93,3318	92,6659	92,7239	92,7145	92,4336	92,3211	92,0592	92,5675
Eight	95,8288	88,4858	95,416	94,9815	95,416	95,5898	95,6985	95,5898	94,9381	95,5247
Fandisk	96,6346	95,4913	95,8003	95,2209	94,845	95,2261	95,4475	95,2982	94,415	95,2776
Feline	94,9143	94,6099	93,6085	93,3428	93,1767	93,1065	92,9958	92,8612	92,7853	92,9618
Foot	94,7295	94,0053	93,4543	92,7584	92,8017	92,7934	92,6086	92,4771	92,0859	92,7385
Geosphere	91,023	54,4885	90,2923	90,1879	92,9019	91,9624	92,9019	91,3361	92,2756	93,0063
Hand	95,8565	95,5977	94,7339	94,9461	94,1884	94,5212	94,5953	94,4868	93,8343	94,3187
Happy	94,6783	94,4327	93,3124	93,4131	92,9815	93,0371	92,7979	92,8699	92,6215	92,6164
Head	97,698	96,9758	97,0613	96,8234	96,2593	96,624	96,9758	96,728	96,1539	96,6396
Horse	94,871	94,4167	93,6197	93,0813	93,0141	92,9192	92,7924	92,6614	92,4691	92,8646
Nefertiti	89,7536	70,8287	88,7458	86,2262	87,626	87,1221	88,1859	87,738	85,1624	88,3539
Statue	94,962	94,7372	93,6302	93,8385	93,2509	93,3871	93,1924	93,3144	92,88	92,9944
Sculpt	94,6924	94,2351	93,4509	92,9292	92,8515	92,8764	92,5573	92,5185	92,4067	92,7312
Torus	99,7092	99,5752	99,6173	99,6356	99,5313	99,5386	99,5322	99,4947	99,4769	99,2483
Venus	93,9952	93,1605	92,6443	91,6946	91,7147	91,7954	91,7087	91,4566	90,9525	91,7893

Table 0.12 EdgeBreaker CLERS only bit per vertex (bpv)

EdgeBreaker Clers bpv	bcm	zpaq	bzip2	lzlib	lzma	zstd	balz	brotli	lzham	libdeflate
Angel	2,0909	2,1908	2,6675	2,6291	2,9708	2,8387	2,8435	2,85	3,157	2,9431
Armadillo	1,8968	1,9964	2,4445	2,403	2,7587	2,5993	2,5543	2,609	3,0363	2,6905
Body	3,6343	7,4374	4,2419	5,4121	5,097	4,962	4,6695	4,8045	6,0647	4,6357
Bunny	3,2129	5,0495	3,8233	4,9639	4,7068	4,4391	4,2838	4,4873	5,3548	4,3695
Cow	2,4408	3,449	2,9532	3,6006	3,5592	3,4105	3,2534	3,4187	3,9835	3,3581
Dinosaur	2,5859	2,8509	3,2	3,5195	3,4917	3,4962	3,631	3,685	3,8107	3,5667
Eight	2,0052	5,5352	2,2037	2,4125	2,2037	2,1201	2,0679	2,1201	2,4334	2,1514
Fandisk	1,6148	2,1634	2,0151	2,2931	2,4735	2,2907	2,1844	2,2561	2,6798	2,2659
Feline	2,4412	2,5874	3,068	3,1956	3,2753	3,309	3,3621	3,4268	3,4632	3,3785
Foot	2,5288	2,8762	3,1406	3,4744	3,4537	3,4577	3,5463	3,6094	3,7971	3,484
Geosphere	4,2469	21,5309	4,5926	4,642	3,358	3,8025	3,358	4,0988	3,6543	3,3086
Hand	1,9889	2,1132	2,5278	2,426	2,7897	2,6299	2,5943	2,6464	2,9596	2,7271
Happy	2,5554	2,6733	3,2113	3,1629	3,3701	3,3435	3,4583	3,4237	3,543	3,5455
Head	1,1047	1,4513	1,4102	1,5244	1,7951	1,6201	1,4513	1,5702	1,8457	1,6126
Horse	2,4611	2,6792	3,0616	3,3199	3,3522	3,3977	3,4586	3,5214	3,6137	3,4239
Nefertiti	4,8963	13,9398	5,3779	6,5819	5,913	6,1538	5,6455	5,8595	7,0903	5,5652
Statue	2,4182	2,5261	3,0575	2,9575	3,2396	3,1742	3,2676	3,2091	3,4176	3,3627
Sculpt	2,5473	2,7668	3,1431	3,3935	3,4308	3,4189	3,572	3,5907	3,6443	3,4886
Torus	0,1396	0,2039	0,1837	0,1749	0,225	0,2215	0,2245	0,2425	0,2511	0,3608
Venus	2,8815	3,2821	3,5298	3,9855	3,9758	3,9371	3,9787	4,0997	4,3416	3,94

Table 0.13 Alliez & Desbrun connectivity only compression ratio

Alliez Desbrun Comp. Ratio	bcm	zpaq	bzip2	lzlib	lzma	zstd	balz	brotli	lzham	libdeflate
Angel	16	16	13	13	13	13	12	12	12	12
Armadillo	15	14	12	12	12	11	11	11	11	11
Body	9	4	8	6	6	6	6	7	5	7
Bunny	9	6	8	6	6	7	7	7	5	7
Cow	13	9	11	9	9	9	10	10	8	10
Dinosaur	11	10	10	8	8	8	8	8	8	8
Eight	37	7	34	28	38	35	37	34	37	38
Fandisk	24	17	18	15	15	16	17	17	13	16
Feline	11	11	10	9	9	9	8	8	8	9
Foot	11	10	10	8	8	8	8	8	7	8
Geosphere	10	2	8	8	12	12	14	12	12	15
Hand	14	14	12	17	11	12	11	11	11	11
Happy	11	11	9	9	9	9	8	8	8	8
Head	58	39	42	40	34	39	43	41	32	38
Horse	11	11	10	8	8	8	8	8	8	8
Nefertiti	7	2	6	5	5	5	6	6	5	6
Statue	11	11	10	9	9	9	8	9	9	9
Sculpt	11	10	10	8	8	8	8	8	8	8
Torus	1709	325	1823	793	960	2279	771	2486	663	720
Venus	9	8	8	7	7	7	7	7	6	7

Table 0.14 Alliez & Desbrun connectivity only storage cost in percentage

Alliez Desbrun Stor. Cost %	bcm	zpaq	bzip2	lzlib	lzma	zstd	balz	brotli	lzham	libdeflate
Angel	6,3833	6,3841	7,809	8,1229	8,1701	8,3252	8,4808	8,5165	8,5655	8,6289
Armadillo	7,0987	7,285	8,5896	9,0453	9,0516	9,2377	9,6569	9,5232	9,5982	9,6693
Body	12,1085	28,9855	13,5577	19,4016	18,2796	17,2978	16,9705	16,0355	22,0196	16,316
Bunny	11,3596	19,2427	12,9318	17,4491	16,8512	15,9876	15,8547	15,3454	20,0177	15,6776
Cow	7,7493	11,7721	9,3789	11,9316	11,3732	11,2365	10,7578	10,5755	13,4131	10,735
Dinosaur	9,3096	10,109	10,7563	12,7606	12,703	12,7744	13,1062	12,8412	14,0945	12,6017
Eight	2,7202	14,81	2,9793	3,6701	2,6339	2,8929	2,7634	2,9793	2,7202	2,677
Fandisk	4,2883	6,2085	5,6268	6,7542	6,7696	6,4865	6,0644	6,0952	7,7477	6,2857
Feline	9,386	9,6712	10,7629	12,3571	12,3345	12,4838	12,913	12,621	13,1189	12,4251
Foot	9,3304	10,5594	10,6889	13,0506	12,9742	13,0506	13,2897	12,9277	14,5752	12,652
Geosphere	10,6996	68,3128	12,7572	13,3745	8,4362	9,0535	7,4074	8,4362	8,8477	6,9959
Hand	7,3637	7,4575	8,7714	6,0676	9,2707	8,3903	9,7927	9,6503	9,661	9,8583
Happy	9,8138	9,8267	11,2328	12,0931	12,0783	12,2472	13,1032	12,6641	12,6506	12,8082
Head	1,7484	2,597	2,4205	2,5515	3,027	2,6056	2,3464	2,4888	3,1779	2,6995
Horse	9,3029	9,954	10,6955	12,6555	12,5985	12,7274	13,1877	12,7776	13,8171	12,5215
Nefertiti	14,8559	50,5543	17,4058	23,8359	20,9534	21,1752	19,2905	18,4035	23,9468	19,0687
Statue	9,4767	9,5623	10,78	11,3804	11,4787	11,6193	12,5378	12,1278	11,9648	12,3055
Sculpt	9,5905	10,1457	10,9437	12,8708	12,7857	12,9419	13,3363	13,061	13,8591	12,7873
Torus	0,0585	0,3082	0,0549	0,1262	0,1042	0,0439	0,1298	0,0402	0,1509	0,139
Venus	11,4183	12,8566	12,7689	16,008	15,7052	15,9602	16,251	15,7291	17,7331	15,5618

Table 0.15 Alliez & Desbrun connectivity only space savings in percentage

Alliez Desbrun Space Sav. %	bcm	zpaq	bzip2	lzlib	lzma	zstd	balz	brotli	lzham	libdeflate
Angel	93,6167	93,6159	92,191	91,8771	91,8299	91,6748	91,5192	91,4835	91,4345	91,3711
Armadillo	92,9013	92,715	91,4104	90,9547	90,9484	90,7623	90,3431	90,4768	90,4018	90,3307
Body	87,8915	71,0145	86,4423	80,5984	81,7204	82,7022	83,0295	83,9645	77,9804	83,684
Bunny	88,6404	80,7573	87,0682	82,5509	83,1488	84,0124	84,1453	84,6546	79,9823	84,3224
Cow	92,2507	88,2279	90,6211	88,0684	88,6268	88,7635	89,2422	89,4245	86,5869	89,265
Dinosaur	90,6904	89,891	89,2437	87,2394	87,297	87,2256	86,8938	87,1588	85,9055	87,3983
Eight	97,2798	85,19	97,0207	96,3299	97,3661	97,1071	97,2366	97,0207	97,2798	97,323
Fandisk	95,7117	93,7915	94,3732	93,2458	93,2304	93,5135	93,9356	93,9048	92,2523	93,7143
Feline	90,614	90,3288	89,2371	87,6429	87,6655	87,5162	87,087	87,379	86,8811	87,5749
Foot	90,6696	89,4406	89,3111	86,9494	87,0258	86,9494	86,7103	87,0723	85,4248	87,348
Geosphere	89,3004	31,6872	87,2428	86,6255	91,5638	90,9465	92,5926	91,5638	91,1523	93,0041
Hand	92,6363	92,5425	91,2286	93,9324	90,7293	91,6097	90,2073	90,3497	90,339	90,1417
Happy	90,1862	90,1733	88,7672	87,9069	87,9217	87,7528	86,8968	87,3359	87,3494	87,1918
Head	98,2516	97,403	97,5795	97,4485	96,973	97,3944	97,6536	97,5112	96,8221	97,3005
Horse	90,6971	90,046	89,3045	87,3445	87,4015	87,2726	86,8123	87,2224	86,1829	87,4785
Nefertiti	85,1441	49,4457	82,5942	76,1641	79,0466	78,8248	80,7095	81,5965	76,0532	80,9313
Statue	90,5233	90,4377	89,22	88,6196	88,5213	88,3807	87,4622	87,8722	88,0352	87,6945
Sculpt	90,4095	89,8543	89,0563	87,1292	87,2143	87,0581	86,6637	86,939	86,1409	87,2127
Torus	99,9415	99,6918	99,9451	99,8738	99,8958	99,9561	99,8702	99,9598	99,8491	99,861
Venus	88,5817	87,1434	87,2311	83,992	84,2948	84,0398	83,749	84,2709	82,2669	84,4382

Table 0.16 Alliez & Desbrun connectivity only bit per vertex (bpv)

Alliez Desbrun bpv	bcm	zpaq	bzip2	lzlib	lzma	zstd	balz	brotli	lzham	libdeflate
Angel	1,5333	1,5335	1,8757	1,9511	1,9625	1,9997	2,0371	2,0457	2,0575	2,0727
Armadillo	1,7042	1,7489	2,0621	2,1715	2,173	2,2177	2,3183	2,2862	2,3042	2,3213
Body	2,9142	6,9761	3,263	4,6695	4,3994	4,1632	4,0844	3,8594	5,2996	3,9269
Bunny	2,747	4,6533	3,1272	4,2195	4,075	3,8661	3,834	3,7108	4,8407	3,7912
Cow	1,8733	2,8457	2,2672	2,8843	2,7493	2,7163	2,6006	2,5565	3,2424	2,595
Dinosaur	2,2977	2,495	2,6547	3,1494	3,1352	3,1528	3,2347	3,1693	3,4786	3,1102
Eight	0,658	3,5822	0,7206	0,8877	0,6371	0,6997	0,6684	0,7206	0,658	0,6475
Fandisk	1,0292	1,49	1,3504	1,621	1,6247	1,5568	1,4554	1,4629	1,8595	1,5086
Feline	2,2594	2,3281	2,5909	2,9747	2,9692	3,0051	3,1085	3,0382	3,158	2,991
Foot	2,2436	2,5391	2,5703	3,1382	3,1198	3,1382	3,1957	3,1086	3,5048	3,0423
Geosphere	2,5679	16,3951	3,0617	3,2099	2,0247	2,1728	1,7778	2,0247	2,1235	1,679
Hand	1,7683	1,7908	2,1064	1,4571	2,2263	2,0148	2,3516	2,3174	2,32	2,3674
Happy	2,374	2,3771	2,7172	2,9253	2,9218	2,9626	3,1697	3,0635	3,0602	3,0983
Head	0,4197	0,6234	0,581	0,6125	0,7267	0,6255	0,5633	0,5975	0,7629	0,648
Horse	2,2399	2,3967	2,5752	3,0471	3,0334	3,0644	3,1753	3,0765	3,3268	3,0149
Nefertiti	3,5853	12,2007	4,2007	5,7525	5,0569	5,1104	4,6555	4,4415	5,7793	4,602
Statue	2,2812	2,3018	2,595	2,7395	2,7632	2,797	3,0181	2,9194	2,8802	2,9622
Sculpt	2,3107	2,4445	2,6367	3,101	3,0805	3,1182	3,2132	3,1469	3,3391	3,0809
Torus	0,014	0,074	0,0132	0,0303	0,025	0,0105	0,0312	0,0097	0,0362	0,0334
Venus	2,7731	3,1224	3,1011	3,8878	3,8142	3,8761	3,9468	3,82	4,3067	3,7794

Table 0.17 Face Fixer connectivity only compression ratio

Face Fixer Comp. Rat.	bcm	zpaq	bzip2	lzlib	lzma	zstd	balz	brotli	lzham	libdeflate
Angel	38	37	29	31	26	26	28	29	25	26
Armadillo	42	41	32	35	28	28	33	31	28	29
Body	20	11	16	14	14	14	15	15	12	15
Bunny	23	15	19	16	16	16	17	17	14	16
Cow	32	23	26	23	21	22	24	23	19	22
Dinosaur	31	28	24	24	20	21	22	22	18	22
Eight	31	13	25	24	24	24	25	26	23	24
Fandisk	48	36	37	36	29	32	35	34	28	32
Feline	33	31	25	26	22	22	25	24	20	23
Foot	31	28	25	24	21	22	23	23	18	22
Geosphere	12	4	10	10	12	11	12	12	11	12
Hand	40	38	31	34	28	28	31	31	27	28
Happy	31	30	24	25	21	22	23	23	20	21
Head	71	54	55	53	39	46	53	49	43	43
Horse	32	30	25	26	21	22	24	24	19	23
Nefertiti	13	6	11	10	10	11	10	11	9	11
Statue	33	32	25	28	24	23	25	26	23	23
Sculpt	31	29	25	25	21	21	23	23	19	22
Torus	498	321	399	404	297	300	267	281	291	166
Venus	28	24	22	21	18	19	20	20	16	19

Table 0.18 Face Fixer connectivity only storage cost in percentage

Face Fixer Stor. Cost %	bcm	zpaq	bzip2	lzlib	lzma	zstd	balz	brotli	lzham	libdeflate
Angel	2,6661	2,7447	3,4963	3,2603	3,9638	3,9731	3,6201	3,566	4,1131	3,8926
Armadillo	2,3974	2,4872	3,2075	2,924	3,5961	3,6653	3,1092	3,2318	3,6507	3,5388
Body	5,0919	9,5757	6,3791	7,4965	7,3833	7,2419	7,058	7,0014	8,4441	7,1004
Bunny	4,3717	6,8291	5,4171	6,3268	6,6458	6,2521	6,1503	6,2453	7,5691	6,2521
Cow	3,1462	4,3496	3,9783	4,4115	4,9445	4,6556	4,2671	4,3771	5,4671	4,5697
Dinosaur	3,3081	3,6186	4,1799	4,2993	5,0361	4,8954	4,6524	4,5878	5,5676	4,7419
Eight	3,3127	7,9144	4,0732	4,2795	4,3052	4,2408	4,0861	3,9701	4,4986	4,215
Fandisk	2,1128	2,778	2,758	2,8537	3,5374	3,1515	2,9185	2,9463	3,6531	3,2025
Feline	3,1	3,242	4,0025	3,9155	4,6788	4,5733	4,146	4,2367	5,1279	4,4801
Foot	3,2334	3,6565	4,1226	4,2493	4,9998	4,7343	4,3861	4,5387	5,6026	4,6635
Geosphere	8,7244	30,3249	10,1083	10,349	9,0253	9,1456	9,0253	8,8448	9,9278	8,7244
Hand	2,5335	2,6578	3,3201	2,9968	3,6976	3,6927	3,3281	3,3188	3,7908	3,6457
Happy	3,2858	3,3811	4,2113	4,0721	4,8153	4,7204	4,3923	4,3763	5,1107	4,7675
Head	1,4102	1,8834	1,8484	1,9108	2,5941	2,2131	1,8903	2,0568	2,3404	2,3643
Horse	3,1362	3,3799	4,0165	3,9979	4,7841	4,6053	4,2467	4,3228	5,2772	4,5056
Nefertiti	7,7346	18,3912	9,5222	10,7941	10,0034	9,8316	10,1753	9,7972	11,2754	9,5909
Statue	3,0886	3,1583	4,0121	3,6435	4,3284	4,4516	4,0019	3,9874	4,5411	4,4566
Sculpt	3,2501	3,4969	4,1284	4,1507	4,9089	4,765	4,35	4,4776	5,4118	4,6807
Torus	0,2011	0,3116	0,2513	0,2477	0,3368	0,3344	0,3752	0,3571	0,3448	0,6054
Venus	3,7027	4,1669	4,6371	4,92	5,6199	5,3539	5,1037	5,2173	6,2654	5,3818

Table 0.19 Face Fixer connectivity only space savings in percentage

Face Fixer Spac. Sav. %	bcm	zpaq	bzip2	lzlib	lzma	zstd	balz	brotli	lzham	libdeflate
Angel	97,3339	97,2553	96,5037	96,7397	96,0362	96,0269	96,3799	96,434	95,8869	96,1074
Armadillo	97,6026	97,5128	96,7925	97,076	96,4039	96,3347	96,8908	96,7682	96,3493	96,4612
Body	94,9081	90,4243	93,6209	92,5035	92,6167	92,7581	92,942	92,9986	91,5559	92,8996
Bunny	95,6283	93,1709	94,5829	93,6732	93,3542	93,7479	93,8497	93,7547	92,4309	93,7479
Cow	96,8538	95,6504	96,0217	95,5885	95,0555	95,3444	95,7329	95,6229	94,5329	95,4303
Dinosaur	96,6919	96,3814	95,8201	95,7007	94,9639	95,1046	95,3476	95,4122	94,4324	95,2581
Eight	96,6873	92,0856	95,9268	95,7205	95,6948	95,7592	95,9139	96,0299	95,5014	95,785
Fandisk	97,8872	97,222	97,242	97,1463	96,4626	96,8485	97,0815	97,0537	96,3469	96,7975
Feline	96,9	96,758	95,9975	96,0845	95,3212	95,4267	95,854	95,7633	94,8721	95,5199
Foot	96,7666	96,3435	95,8774	95,7507	95,0002	95,2657	95,6139	95,4613	94,3974	95,3365
Geosphere	91,2756	69,6751	89,8917	89,651	90,9747	90,8544	90,9747	91,1552	90,0722	91,2756
Hand	97,4665	97,3422	96,6799	97,0032	96,3024	96,3073	96,6719	96,6812	96,2092	96,3543
Happy	96,7142	96,6189	95,7887	95,9279	95,1847	95,2796	95,6077	95,6237	94,8893	95,2325
Head	98,5898	98,1166	98,1516	98,0892	97,4059	97,7869	98,1097	97,9432	97,6596	97,6357
Horse	96,8638	96,6201	95,9835	96,0021	95,2159	95,3947	95,7533	95,6772	94,7228	95,4944
Nefertiti	92,2654	81,6088	90,4778	89,2059	89,9966	90,1684	89,8247	90,2028	88,7246	90,4091
Statue	96,9114	96,8417	95,9879	96,3565	95,6716	95,5484	95,9981	96,0126	95,4589	95,5434
Sculpt	96,7499	96,5031	95,8716	95,8493	95,0911	95,235	95,65	95,5224	94,5882	95,3193
Torus	99,7989	99,6884	99,7487	99,7523	99,6632	99,6656	99,6248	99,6429	99,6552	99,3946
Venus	96,2973	95,8331	95,3629	95,08	94,3801	94,6461	94,8963	94,7827	93,7346	94,6182

Table 0.20 Face Fixer connectivity only bit per vertex (bpv)

Face Fixer bpv	bcm	zpaq	bzip2	lzlib	lzma	zstd	balz	brotli	lzham	libdeflate
Angel	2,133	2,1959	2,7973	2,6084	3,1712	3,1787	2,8963	2,853	3,2907	3,1143
Armadillo	1,918	1,9898	2,5661	2,3393	2,8769	2,9323	2,4874	2,5855	2,9206	2,8311
Body	4,0506	7,6174	5,0745	5,9634	5,8734	5,7609	5,6146	5,5696	6,7173	5,6484
Bunny	3,4485	5,3869	4,2731	4,9906	5,2423	4,9317	4,8514	4,9264	5,9705	4,9317
Cow	2,5207	3,4848	3,1873	3,5344	3,9614	3,73	3,4187	3,5069	4,3802	3,6612
Dinosaur	2,6473	2,8958	3,345	3,4405	4,0301	3,9176	3,7231	3,6714	4,4554	3,7947
Eight	2,6841	6,4125	3,3003	3,4674	3,4883	3,436	3,3107	3,2167	3,6449	3,4151
Fandisk	1,6914	2,2239	2,2079	2,2845	2,8318	2,5229	2,3364	2,3586	2,9245	2,5637
Feline	2,4805	2,5941	3,2026	3,133	3,7438	3,6594	3,3175	3,39	4,1032	3,5848
Foot	2,5879	2,9265	3,2995	3,401	4,0016	3,7891	3,5104	3,6326	4,484	3,7324
Geosphere	7,1605	24,8889	8,2963	8,4938	7,4074	7,5062	7,4074	7,2593	8,1481	7,1605
Hand	2,0269	2,1264	2,6563	2,3976	2,9583	2,9544	2,6627	2,6552	3,0329	2,9168
Happy	2,63	2,7064	3,3708	3,2594	3,8543	3,7783	3,5157	3,503	4,0907	3,816
Head	1,1286	1,5073	1,4793	1,5292	2,076	1,7712	1,5128	1,6461	1,873	1,8922
Horse	2,5095	2,7045	3,2139	3,199	3,8281	3,6851	3,3981	3,459	4,2227	3,6053
Nefertiti	6,0201	14,3144	7,4114	8,4013	7,786	7,6522	7,9197	7,6254	8,7759	7,4649
Statue	2,4709	2,5267	3,2097	2,9148	3,4627	3,5613	3,2015	3,1899	3,6329	3,5653
Sculpt	2,6006	2,7981	3,3034	3,3213	3,9279	3,8128	3,4807	3,5828	4,3303	3,7453
Torus	0,1609	0,2493	0,201	0,1982	0,2695	0,2675	0,3002	0,2858	0,2759	0,4844
Venus	2,9637	3,3353	3,7117	3,9381	4,4983	4,2854	4,0851	4,1761	5,015	4,3077