A Topological Feature Heuristic for Subdivision Model Inspection and Analysis

by

Christopher Taylor

Bachelor of Information Technology, Carleton University, 2010

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs

in partial fulfillment of the requirements for the degree of
Master of Science in Information and Systems Science

Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, Canada, K1S 5B6
January 2013

© Copyright 2013, Christopher Taylor

Library and Archives
Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 978-0-494-94306-9

Our file Notre référence

ISBN: 978-0-494-94306-9

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

Subdivision modeling is a common tool when creating high polygon count models which relies on proper graph topology as well as subdivision algorithms to yield high detail wherever required in specific areas on the model. These models are increasingly becoming more prominent in the development cycles and asset modifications stages in the entertainment industry. However, despite there being numerous compression algorithms which target triangulated models, there is a need to analyze high-polygon subdivision models to compress the information stored. In this thesis we present a novel heuristic method of capturing the underlying topological structure of subdivision models based solely on their graph connectivity data. Our method is quick and robust to changes in geometry, without compromising general accuracy in the area of geometric compression.

Acknowledgements

I would like to thank my supervisor, Professor Chris Joslin, for his guidance throughout the development of this work. Thanks to my parents who have encouraged and supported me in undertaking this academic experience. I am also very grateful to all my colleagues in the lab, Sina Firouzi, Parnia Farokhian, and Junior, for all of their academic discussions and pleasant company in the lab. Finally, this thesis would not have been possible without the support, advice and patience from my fellow colleague Omar Hesham.

Table of Contents

Abstract.....	iii
Acknowledgements	iv
List of Tables	viii
List of Figures.....	ix
List of Acronyms	xii
Chapter 1 Introduction.....	1
1.1 Procedural Model Description	1
1.2 Motivation and Problem Description	5
1.3 Research Contributions	6
1.4 Thesis Overview	7
Chapter 2 Background Information	8
2.1 3D Polygon Mesh Models.....	8
2.1.1 Common Terms and Definitions	8
2.1.2 Mesh Representations.....	10
2.1.3 Subdivision modeling	12
2.2 Compression	15
2.2.1 Common Terms and Definitions	16
2.2.2 Lossless methods	16
2.2.3 Entropy	17
2.3 3D Compression.....	18

2.4	Shape Analysis.....	20
2.5	Topology	25
2.5.1	Mesh topology	27
Chapter 3	Literature Review	29
3.1	3D Compression.....	29
3.1.1	Triangle based methods	30
3.1.2	Non-triangle based methods	37
3.1.3	Degree-valence approach.....	38
3.1.4	Geometry coding	38
3.1.5	Prediction.....	39
3.1.6	Vector Quantization.....	40
3.2	Shape Analysis.....	41
3.2.1	Structure Detection	42
3.2.2	Symmetry detection	43
3.2.3	Reverse engineering.....	45
3.2.4	Inverse procedural modeling	47
Chapter 4	Method	49
4.1	Macro Decomposition.....	52
4.2	Micro Decomposition	53
4.2.1	Extraordinary Vertices.....	54
4.2.2	Curves	54
4.2.3	Surface Patches	61
4.3	File Format.....	64
4.4	Re-creation.....	66

4.4.1	Coons interpolation.....	68
4.5	Approximation Control.....	68
4.6	Spatial Analysis.....	71
Chapter 5	Results.....	77
5.1	Micro Results.....	78
5.1.1	File Compression.....	78
5.1.2	Underlying topology structure.....	78
5.2	Recreation	88
5.3	Observation and Results.....	96
Chapter 6	Conclusion	99
6.1	Conclusions.....	99
6.2	Future Research Directions.....	100
References		101
Appendix A - Example MPD document		112

List of Tables

Table 1: Description of common mesh representations.....	11
Table 2: Subdivision scheme classification.	14
Table 3: Polygon cylinder caps and their resultant underling topological structure.....	60
Table 4: File Compression Results	78
Table 5: Recreation results.....	89

List of Figures

Figure 1.1: Diagram of three geometrically-different primitives with the same intrinsic topology.	3
Figure 2.1: Arrangement of consistent patch shapes of a) triangles b) quadrilaterals c) hexagons.	9
Figure 2.2: Illustration of two types of extraordinary vertices in a quadrilateral polygon mesh.	13
Figure 2.3: Example of Catmull-Clark subdivision process [MAYA].	15
Figure 2.4: Diagram of connectivity information and geometry	19
Figure 2.5: Diagram of range query on a set of 2D points.	22
Figure 2.6: Diagram of nearest neighbor query.	23
Figure 2.7: Diagram of k-Nearest neighbor query.	24
Figure 2.8: Example of special cases of Minkowski distance a) L_1 Manhattan distance, b) Euclidean distance c) L_6 , d) L_∞ maximum (infinity) distance $L_\infty = \max_{i=1}^n x_i - y_i $	24
Figure 2.9: Example of Hausdorff calculation.	25
Figure 2.10: Example classification of manifolds.	26
Figure 2.11: Examples of good topologies in organic and non-organic models.	28
Figure 3.1: Example triangle strips.	30
Figure 3.2: Chow's triangle strip.	31
Figure 3.3: Diagram of spanning tree.	32

Figure 3.4: Diagram of layered decomposition.	33
Figure 3.5: Example of the processed used in the Edgebreaker algorithm.....	36
Figure 3.6: Excerpt of 3D model database retrieval flow diagram.....	42
Figure 4.1: Visual diagram of model reconstruction algorithm.....	51
Figure 4.2: Breakdown of micro decomposition algorithm.....	54
Figure 4.3: Curve component breakdown and the various functions involved in the micro decomposition phase.	56
Figure 4.4: Occurrence of extraordinary vertices resulting from extrusion.....	57
Figure 4.5: Simple patch description.	61
Figure 4.6: Patch type classification for surface patches.	62
Figure 4.7: Diagram of the right turn concept based upon the surface normal direction.	63
Figure 4.8: Patch filling reference chart.	67
Figure 4.9: Diagram of Coon's algorithm.	68
Figure 4.10: Diagram of approximation control on a simple patch.....	69
Figure 4.11: Example issue with reconstruction and proposed solution with additional curve information stored based on geometric change along a patch axis.....	71
Figure 4.12: Examples of discrete transformations seen in meshes (a and b) and surface patches (c and d) of linear type (a and c) and rotational (b and d)	72
Figure 4.13: Visual storyboard of spatial analysis process.....	75
Figure 4.14: Diagram of shape analysis construction process for the rotational and linear occurrences.	76

Figure 5.1: Topological analysis for model (Bottle cap).	80
Figure 5.2: Topological analysis for model (Bottle).....	81
Figure 5.3: Topological analysis for model (Can).	82
Figure 5.4: Topological analysis for model (Misc. Shape 02).....	83
Figure 5.5: Topological analysis for model (Misc. Shape 03).....	84
Figure 5.6: Topological analysis for model (Chair 02).....	85
Figure 5.7: Topological analysis for model (Chair 03)......	86
Figure 5.8: Topological analysis for model (Fire hydrant).....	87
Figure 5.9: Bottle cap model, both original and recreated result.	90
Figure 5.10: Bottle model, both original and recreated result.	90
Figure 5.11: Can model, both original and recreated result.....	91
Figure 5.12: Chair 02 model, both original and recreated result.	92
Figure 5.13: Chair 03 model, both original and recreated	92
Figure 5.14: Misc. shape 02 model, both original and recreated result.	93
Figure 5.15: Misc. shape 03 model, both original and recreated result.	94
Figure 5.16: Fire hydrant model, both original and recreated result.	95

List of Acronyms

Acronyms	Definitions
BFS	breadth first search
DFS	depth first search
BPV	bits per vertex
CW	clockwise
CWW	counter-clockwise

Chapter 1

Introduction

1.1 Procedural Model Description

As the video game industry progresses and modern games constantly improve their resolution and visual fidelity, there is an increasing need for efficient encoding and compression schemes on both sides of the product. On the development side, studios tend to favor modeling practices and tools that allow high compression storage, efficient editing, and an easier means of transferability to other programs/tools. On the consumer's end, the game is expected to deliver these increasingly complex virtual environments in real-time performance, raising the need for run-time optimization of a scene's models (e.g. Level of Detail and progressive mesh streaming). The performance of the optimization of these methods can be limited if proper modeling and design practices are not followed and maintained.

A critical factor in achieving this notion of efficiency is a clean and well maintained surface topology, which are the connections between vertices and edges that form the final polygonal shape. This is a common concept in high-polygon models which relies on subdivision algorithms such as Catmull-Clark [Sal06]. One of the major components of the modeling process is having good topology, namely the flow, form, and density of the arrangements of polygon faces. The concept is demonstrated here using simple primitive models as an example in Figure 1.1. The sphere, cube and the cylinder all have the same underlying topological graph structure. Their differences lie in their geometric position and mesh resolution. Two identical sets of vertices can form geometrically similar models, but one can be more “topologically optimized” than the other or even contain less vertices to describe the same geometric shape. This is why topology is arguably more important than mesh resolution in subdivision modeling, because if the underlying topology is lacking or if the mesh had been incorrectly modeled; any subsequent layers of detail would inherit such problems. The concept of subdivision modeling and topology is visited again in more detail in Chapter 2.

Figure 1.1: Diagram of three geometrically-different primitives with the same intrinsic topology.

Similarly, models can also be constructed based on a series of heuristics (rules) and mathematical equations found in procedural modeling. This is a technique that relies on operations based upon mathematical equations, rules, and conditions used to define and create entire 3D objects models. Its mathematical basis, reproducibility and its tiny footprint (requiring only a minimal set of equations and rules/conditions) made it the de-facto modeling paradigm for industrial and engineering applications such as NURBS modeling used in CAD applications. Conversely, methods which analyze models break down its key components into a set of instructions and descriptive actions or equations are found in reverse engineering and inverse procedural modeling.

However unlike in the structured approach of asset generation with procedural modeling

and the analysis approaches of inverse procedural modeling and reverse engineering, these utilize more commonly a destructive modeling approach, whereas operations on geometric information are usually manually defined where in its stored information. This approach is mainly chosen for its creative control and abstract freedom which allows other 3D systems to interact with such model data. The disadvantages to this include issues such as increased file size, and unintuitive editing patterns. Procedural and reverse engineering concepts are further covered in Sections 3.2.4 and 3.2.3.

This notion of efficient topology can further be extended into game development processes, which favor quicker alterations, modifications and more responsive and intuitive controls. Tools such as the edge loop tool and the extrusion tool help users modify existing objects such as a primitive cube into visually pleasing and geometrically complex objects such as a full humanoid face in a much faster development timeline.

Our work focuses on a 3D model's underlying topological structure, its graph topology information where we store and transfer relevant mesh information then recreate the compressed model on the screen when needed. This is designed for both memory and network constrained devices. We store these 3D objects in memory using our heuristical graph-based approach. The models can also be stored to disk using our custom file format. While a perfect reconstruction is ideal, the primary goal of our work is to first retain and quickly present the topology of the model, after which the full model can be optionally loaded if resources permit.

1.2 Motivation and Problem Description

We set out to develop a method to extract procedural modeling data involving equations, rules, and constraints that would allow us to:

- Recreate polygonal models. A means of inverse procedural modeling that reverse engineered the equations and rules required to rebuild a model. This concept would have given us a powerful and efficient means of *compressing* the model based upon its required rule set.
- Provide a *visual abstraction and summary* of the model's main defining features.
- Acquire a sense to observe and possibly even judge the *topological quality* of its construction.

Existing methods [Shi01] that tackle this problem provide a high level of accuracy and visual fidelity but at the cost of implementation and processing time. Their results are also dependent upon geometric features of a given model (such as their surface normal).

Our efforts focus on 3D subdivision polygon models, commonly found in both the game and film industry. With the above goals in mind, we set out to develop an efficient heuristic that can capture a model's overall design that relies on its *graph* representation; we wanted to retrieve the best possible set of procedural descriptors of the model's

overall design. We do this by taking advantage of inherit subdivision model properties as well as a few other practical subtleties which we've incorporated into an efficient heuristic analysis tool.

1.3 Research Contributions

This thesis presents our research efforts towards analyzing a given subdivision model into surface patches that retain the original topology that is intended for both organic and man-made models. With these motivations in mind, we provide the following results:

- **Novel heuristic graph-based analysis technique of subdivision polygon meshes models.**
- **Ability to accurately describe 3D objects with up to 95% loss-less compression.**

- **Ability to analyze and detect redundant features within a given mesh.**

Especially in non-organic models, the concept of symmetry is highly common.

Our method analyzes for common patches and reduces the needed information to describe such features within a given mesh.

- **Ability to provide a level of approximation which would control the amount of lossy resolution kept in the described model.** The ability to increase or decrease image resolution is an important ability when dealing with matters of compression. Our method describes of a process that would allow for user-based

control over the amount of increased or decreased detail to describe a given mesh model.

1.4 Thesis Overview

This thesis is organized into several chapters. Chapter 1 introduces the topic and describes the motivation, goals and contributions of the thesis. Chapter 2 follows giving a background overview of mesh representation, general compression and shape analysis. Chapter 3 gives a theoretical review of 3D compression and shape analysis. The method is detailed in Chapter 4, followed by a discussion of the results in Chapter 5. Chapter 6 concludes with an overview of our contributions along with a brief discussion of the future potential derived from the work completed in this thesis.

Chapter 2

Background Information

In this section we present background information of what is discussed throughout this paper. This section is organized into three main areas: 3D Polygon Mesh Models where common terms are looked at, common data representations, and a general classification of polygon mesh models. Then we will briefly discuss the concept of topology. We will also go over compression schemes including terms and fundamentals methods used as a basis for understanding the 3D polygonal compression techniques described in Section 3.

2.1 3D Polygon Mesh Models

2.1.1 Common Terms and Definitions

A 3D polygon mesh model is composed of 3 general cases

- i. Geometric (Locational)
- ii. Topological (Connectivity)
- iii. Attributes (texture coordinates, normals)

A given 3D polygon mesh model O , is made up of S polygon meshes and any non-geometric properties. A polygon mesh $s \in S$ is composed of a set of V vertices, a set of E edges and a set of P polygons. Vertices are 3 dimensional vectors of point positions from R^3 . Edges represent a pair of references (v_1, v_2) to the list of vertices and polygons are represented by a sequence $(v_1, v_2 \dots v_k)$ of three or more vertices. Usually most vertices in any one polygon will be coplanar. Geometric information of a 3D polygon mesh model relates to the position of its elements (e.g. vertices) commonly in 3D space whereas topological information relates to how these elements are connected together (e.g. vertices connected to faces and edges).

If the mesh is to be considered only having one type of face, there exists 3 possible face consistency types; a triangle (vertex degree of 6), a quadrilateral (vertex degree of 4), or a hexagon (vertex degree of 3). Considering how hexagons are generally composed of several triangles and are rarely used as the underlying face type of a 3D polygon model, we will only discuss the triangle and quadrilateral patches.

Figure 2.1: Arrangement of consistent patch shapes of a) triangles b) quadrilaterals

c) hexagons.

2.1.2 Mesh Representations

The most common mesh representation is the face-vertex which utilizes a shared list of vertices which contains vertex coordinates that stores its geometry. Each face contains a list of vertex indices which store mesh connectivity where the edges are implied.

There are also other data structures which are used for various applications for accessing and manipulating polygon mesh. Their major aim is to facilitate the fast traversal of elements. The following table briefly outlines several variations of mesh representations.

Table 1: Description of common mesh representations

Face-Vertex	<p>The object is represented by a set of faces and vertices and is one of the most commonly used formats where the edges are implicit.</p>
Winged-Edge	<p>This form explicitly represents each of the three elements (vertices, faces and edges). One major drawback is that it requires large storage and increased complexity when addressing edge traversal. Each edge points to exactly</p> <ul style="list-style-type: none"> • 2 polygons • 2 vertices • 4 edges (CW/CCW)
Half-Edge	<p>Similar to winged edge with the exception uses half the edge traversal used lean regular variant for 2-manifolds.</p>

Vertex- Vertex	Where the object is represented by a set of vertices connected to other vertices and is considered one of the simplest representations. Although it's not commonly used since the face/edge information is implicit and is necessary to traverse data to generate list of faces.
-------------------	--	---

2.1.3 Subdivision modeling

Subdivision modeling is a common practice in the 3D modeling community in games, film and 3D animation. Part of its origin was in response to address arbitrary topology modeling for splines which can be seen in examples from the famous Pixar short, Geri's game [De98]. The basic underlying concept behind subdivision is that a smooth surface can be represented by the limit of a sequence of successive refinements [Zor00]. The final smooth shape is generated based upon a coarse polyhedral mesh which the process is performed upon using a recursive process that subdivides a mesh recursively until a desired level of smoothness is achieved.

In any given subdivision model with a consistent polygon sizes, there will exist vertices that are not of common degree size. Such vertices are called “extraordinary vertices”. In quadrilateral patches they are commonly referred to as poles where E-Poles and N-Poles represent vertices of degree 5 and 3 respectively. Our work will briefly refer to these pole arrangements based on their vertex degrees in Section 4.2.2. An example below

illustrates two types of extraordinary vertices, one of 6 (high) and 3 (low) in a quadrilateral polygon mesh.

Figure 2.2: Illustration of two types of extraordinary vertices in a quadrilateral polygon mesh.

In the case of subdivision schemes, there are many available, each scheme is designed towards a different process in mind, however most schemes are based upon 4 criteria:

- Refinement type (face or vertex split)
- Generated mesh type (triangle or quadrilateral)
- Approximating or interpolating

The following table is recreated from [Zor00] and outlines some of the key subdivision schemes, organized by the previously mentioned criteria.

Table 2: Subdivision scheme classification.

	Face Split		Vertex Split
	Triangle mesh	Quadrilateral Mesh	
Approximating	Loop	Catmull-Clark	Doo-Sabin, Mid-edge
Interpolating	Mod. Butterfly	Kobbelt	Biquartic

Since the intended goal for this work is targeted for the 3D community and that one of the more popular 3D modeling software is Maya [MAYA], the Catmull-Clark subdivision scheme will be explained here. For more comprehensive look at subdivision schemes see [Sal05] and [Zor00].

The Catmull-Clark subdivision scheme is essentially an extension to arbitrary polygonal surfaces. It subdivides a given polygonal mesh by generating new faces, edges and vertices and connects them. The points follow a simple set of rules to be generated:

- Face points, which are of the original mesh (previous iteration) and are calculated as the average of all incident vertices to the current face
- Edge points, where each of the interior edges of the polygon surface and are calculated based on the average of each edge midpoint and the two face points that lie adjacent to the current edge.
- Vertex points, that is for each interior vertex P of the original mesh. It is the average of Q , the average of adjacent face points to P , $2R$, where R represents the average of midpoints of incident edge on P and $S\frac{n-3}{4}$ which represents P itself.

Once generated the face and vertex points are connected in the following way. Each face

point is connected to all edge points of the interior edges bounding its face. Each vertex point is connected to all edge points used in its calculation. What's powerful about this method is that it is the resultant subdivision will guarantee a polygon consistency of only quadrilaterals, despite possibly having faces of arbitrary degree in its control polyhedron. Figure 2.4 illustrates the subdivision process to a cube and a torus primitive.

Figure 2.3: Example of Catmull-Clark subdivision process [MAYA].

2.2 Compression

This section provides a brief review on compression including common terms and methods used. The main concept behind compression is the notion of reducing redundancy found in source of data [Sal09]. The general law of data [Sal09] assigns short codes to more common events (symbols/phrases) and longer codes to rarer events presents.

2.2.1 Common Terms and Definitions

Symmetrical & asymmetrical compression. This occurs when the compressor and decompressor uses the same algorithm but in reverse. Asymmetrical compression design is when either the compressor or decompressor works harder than the other. This asymmetric design is the most commonly used type of compression, where generally the compressor will work harder.

2.2.2 Lossless methods

Lossless methods allow the exact original data to be reconstructed from that of the compressed data. It is commonly used in applications such as the ZIP format and is used in cases when it is important that the original and decompressed data is to be identical in such data sources as those found in text documents. It measures the effect of compression based on the amount of how much the source file has shrunk compared to the original source.

$$CR = \frac{\text{size after compression}}{\text{size before compression}} \quad (2.1)$$

$$\frac{\text{size before} - \text{size after}}{\text{size before}} \% \quad (2.2)$$

2.2.3 Entropy

When it comes to encoding there are two main goals kept in mind, to preserve the underlying information and to minimize the storage required to represent such information. In information theory, entropy is the measure of uncertainty associated with random variables. More formally it is the mean (expected) amount of information on S , the event set or alphabet of the information. This is commonly known as Shannon's Entropy which can be represented by the equation below. The entropy, $H(P)$ represents the probability of occurrences of the symbols of the set of elements.

$$H(P) \equiv - \sum_{j=1}^N p_j \log p_j \quad (2.3)$$

This represents the absolute limits or the best possible lossless compression used to tie in the data encoded with the expected maximum compression ratio. In data compression, the objective for entropy encoding is to achieve an optimal entropy rate for a given set of elements and is the best possible lossless compression rate.

In this review we will discuss several common forms of entropy coding, RLE, used in repetitive sequences and two common statistical methods, Huffman and Arithmetic compression methods.

Run Length Encoding (RLE). It is considered one of the simplest forms of data

compression in which runs of data are compressed into a tuple codeword $\langle r, l, s \rangle$ representing the run flag, the number of consecutive appearances and the symbol respectively.

2.3 3D Compression

Early 3D mesh compression focused on single-rate compression techniques in order to save bandwidth between CPU and graphics cards. In a single-rate compression algorithm all of its data (connectivity and geometry) are compressed and decompressed as a whole. It wasn't until later when the internet existed that progressive compression and transmission began to become heavily researched. Unlike in single-rate compression, in progressive compression algorithms a 3D mesh is reconstructed in various levels of detail (LOD) from course to fine. Furthermore in single-rate algorithms which require the content to be transmitted as a single block, in progressive the transmission can be stopped at any point during the process while still being able to reconstruct the model.

Typically 3D meshes can be represented in two or three categories; connectivity, geometry, attributes. In the early 1990s, it was common for models to be triangular meshes because they proved to be an effective representation. Their vertex description doesn't alter its overall geometry and it allows for efficient mathematical processing since a planar intersection can be created from all three vertices of its triangular face. Connectivity data describes the adjacency relationship between elements which make up a model, typically the incidence between elements that are implied by the meshes' topology. The valence of a vertex describes the number of incident edges to the vertex

and the degree of a face describes the number of edges incident to it. A diagram illustrating these two forms of connectivity can be seen in Figure 2.4.

Figure 2.4: Diagram of connectivity information and geometry

Geometry data specifies geometric information such as the vertex locations in $\mathbb{R}^3$.

Property data specifies other attributes associated with a model such as its texture coordinates, material information and normal vectors.

The importance of 3D mesh compression has been so great that it has also been

incorporated into several international standards. The following is a list of some international standards incorporating 3D mesh compression.

- VRML, the standard for transmitting 3D models across the internet [VRML97] which was originally in ASCII without compression but later incorporated a compressed binary format based on a topological survey algorithm by Taubin et al [Tau98].
- MPEG-4, an ISO/IEC multimedia standard developed by the Motion Picture Experts Group [MPEG4-V01] which included 3DMC three-dimensional mesh coding algorithm which is a coder for manifold triangular meshes based on the topological survey algorithm.

2.4 Shape Analysis

3D objects have a digital representation of their form (shell, volume point). In order to make an adequate comparison, a simplification of the object's representation is required. This simplified representation is called a shape descriptor. Essentially a shape descriptor conveys the most important information of an object. Usually it is invariant to transformation congruency while maintaining efficient storage and handling of its information. Generally these descriptors will establish the following:

1. Normalization attributes (translation, scale and orientation)
2. Generate shape descriptor
3. Compute comparisons (distance) between descriptor pairs

Normalizing attributes (pose normalization) addresses how usually 3D models come with

no prior knowledge and have arbitrary translation, scale and orientation. This process normally is dependent upon calculating a mode's center of mass which calculates the average of all the surface points, resulting in a centroid of a given model. The center of mass is represented by

$$C = \frac{1}{n} \sum_{i=1}^n v_i \quad (2.4)$$

$$S = \sqrt{\frac{1}{n} \sum_{i=1}^n \|v_i\|^2} \quad (2.5)$$

The translation attribute is normalized by translating the center of mass to the origin. The scale is normalized by the average distance of surface points to the center of mass. Unlike the other two channels, rotational is usually normalized by applying the Principal component axis method [Abd10]. This process aligns the principal axis to the X, Y, Z axis based on an affine transformation where rotations are applied in order of the largest variance.

Matching determines the similarity between two shapes. The most common method is achieved by calculating distances. Another way can be done by indexing which generates data structures to speed up search time. Dissimilarity is the measure of similarity between two objects which calculates the distance between two shape descriptors

$\delta: \text{OBJ} \times \text{OBJ} \rightarrow \mathbb{R}^+$ where small values of δ represents large similarity between two objects. Two common types of similarity queries are the range query and k-nearest

neighbor.

Range query

The range query is a search algorithm which given an input distance returns all elements that are within the given distance from the query source point. This is represented in the Equation 2.6 and illustrated in Figure 2.5.

$$R(q, r) = \{x \in X \mid d(q, x) \leq r\} \quad (2.6)$$

Figure 2.5: Diagram of range query on a set of 2D points.

Nearest-Neighbor Query

The nearest neighbor query is similar to the range query where given a source element, finds the closest point to the source element. This is represented in Equation 2.7 and illustrated in Figure 2.6.

$$NN(q) = x \quad (2.7)$$

$$d(q, x) \leq d(q, y) \quad x \in X, \forall y \in X \quad (2.8)$$

Figure 2.6: Diagram of nearest neighbor query.

Similarly the k-nearest neighbor query which in turn looks for k number of elements closest to a source element. This is represented in the Equations 2.9 – 2.11 and illustrated in Figure 2.7.

$$k - NN(q, k) = A \quad (2.9)$$

$$A \subseteq X, |A| = k \quad (2.10)$$

$$d(q, x) \leq d(q, y) \quad \forall x \in A, \forall y \in X - A \quad (2.11)$$

Figure 2.7: Diagram of k-Nearest neighbor query.

Minkowski Distance

This is the metric in Euclidean space which is defined on n dim vectors. This is represented in the following Equation 2.12 and illustrated Figure 2.9.

$$L_p[(x_1, x_2, \dots, x_n)(y_1, y_2, \dots, y_n)] = p \sqrt[p]{\sum_{i=1}^n |x_i - y_i|^p} \quad (2.12)$$

Figure 2.8: Example of special cases of Minkowski distance a) L_1 Manhattan distance, b) Euclidean distance c) L_6 , d) L_∞ maximum (infinity) distance $L_\infty =$

$$\max_{i=1}^n |x_i - y_i|.$$

The Hausdorff Distance

The Hausdorff distance is a distance measure for sets which compares elements by a distance d_s . This measure the extent to which each point of the "model" set A lies near some point of the "image" set B and vice versa. This is represented in equation 2.13 and illustrated in 2.9.

$$d(A, B) = \max\{ d_s(A, B), d_s(B, A) \} \quad (2.13)$$

Figure 2.9: Example of Hausdorff calculation.

2.5 Topology

Homeomorphism. This is when two topological spaces (geometric object) are topologically the same. A common example of this concept is that a square and a circle are homeomorphic whereas a sphere and a donut are not.

Manifolds. A mesh is said to be a manifold or 2-manifold if for every vertex v the faces incident on v are homeomorphic to a disk or semi-disk if v is on a boundary. The manifold mesh will thus be locally planar at all vertices. For every polygon each of its edges are shared by no more than 2 polygons. Examples of different types of manifolds are illustrated in Figure 2.10.

Figure 2.10: Example classification of manifolds.

Connected Components. When two adjacent polygons share an edge then there exists an immediate path. There also exists a path between any two polygons in a model if there is a sequence of adjacent polygons between them, a mesh can be decomposed into connected components if and only if each polygon is adjacent to the other. It should be noted that there may be multiple connected components present in a mesh. This can be done with a simple labeling algorithm, either a BFS (breadth first search) or DFS (depth first search) done in a complexity time of $O(n)$.

Topology. This is the study of the connective properties of objects when these properties are relatively similar when homeomorphism is applied. More specifically in 3D modeling terms, it represents the flow and organization of groups of polygons in a given model based upon their occupying space, in order to organize both the structure and flow of the model's components, edges, vertices and polygons relative to each other.

2.5.1 Mesh topology

A model that contains good topology will be able to deform more. A model that contains good topology would have appropriate areas of density that focuses on several key characteristics:

- **Flow:** Vertices should be equidistant to one another based on the respective areas of a given model
- **Form:** The overall shape should be represented clearly based upon its edges, preserving its form
- **Density:** The curvature and motion of the surface should vary based the surrounding features.

Conversely areas of poor topology would have overly dense mesh with poor flow and lack of sufficient contours that efficiently represent the model's shape and form. The following figure, Figure 2.11 illustrates examples of good topologies for both organic and non-organic 3D subdivision polygon mesh models.

Organic

Non - Organic

Figure 2.11: Examples of good topologies in organic and non-organic models.

Chapter 3

Literature Review

3.1 3D Compression

3D meshes have been widely used in graphics applications for approximating and representing practically any object. However, when representing complex objects or scenes, such data in its raw form and requires significantly large amount of space for storage. Especially since the emergence of the internet the motivation and need for finding query and efficient ways of transmitting such types of information has increased. This has led to an increase in research towards finding better algorithms and data schemes to compress these datasets. In this section we first discuss several common terms and definitions, discussing the general areas of 3D compression. We then we review the key methods from both connectivity and geometry single rate compression algorithms.

3.1.1 Triangle based methods

Early attempts at 3D mesh compression involved dividing the mesh into long strips of triangles, which took advantage of reducing face description based upon shared vertices in adjacent triangles along the strip as seen in Figure 3.1. This reduced data transmission between the CPU and graphics card because less storage space and transmission bandwidth than that of the original index face set of the mesh; however, the amount of compression achieved in this method wasn't significantly high [Pen05].

The concept of a generalized mesh was first introduced by Deering [Dee95] which is composed of general triangle strips with a vertex buffer. The vertex buffer is of FIFO (First In First Out) type and consists of the recently visited vertices. The buffer index requires fewer bits than the buffer of the global vertex index. Based on the assumption that each vertex is reused by the buffer index once, Taubin and Rossignac [Tau98] showed that this method requires approximately 11 bit per vertex (BPV) to encode the connectivity. It should be noted Deering did not propose a method to decompose a mesh into these triangle strips, only the strips themselves.

Figure 3.1: Example triangle strips.

Another method was later proposed based on this work which was a mesh compression scheme that was optimized for real-time rendering by Chow [Cho97]. Chow's method is based upon first finding a set of boundary edges, which it then locates a triangle fan around vertices that are incident on two consecutive bound edges. These combined triangle fans from the first generalized strips are then marked as discovered. Vertices in the previous generalized triangle strip are stored in the buffer in order for the subsequent strip to reuse previously stored vertices. Its effectiveness is optimal only if a triangle mesh is decomposed into long triangle strips. Optimal triangle strip decomposition has been addressed in [Ark96, Eva96A] and heuristics have also been developed for sub-optimal results in [Eva96B, Spe97, Xia99].

Figure 3.2: Chow's triangle strip.

Spanning trees

Turan [Tur84] observed the connectivity of a planar graph can be encoded with a constant number of BPV using two specific spanning trees; a vertex and a triangle spanning tree shown in Figure 3.3. It was then Taubin and Rossignac [Tau98A] who

presented the topologic survey approach to encode the mesh connectivity at specific sets of edges to make a planar polygon. The two spanning trees are run-length encoded where the run represents the tree segment between nodes with degree that is not two, and for each run of the vertex spanning tree the encoder stores two additional flags. The first is the branching bit where a subsequent run would start from and the second is a leaf bit indicating whether a run ends at a leaf node. The triangular spanning tree is always binary and the encoder records matching patterns with 1 bit per triangle index to indicate how to triangulate the internal planar polygon. This algorithm cannot handle non-manifold meshes and requires modifications to handle general manifold meshes of arbitrary genus, meshes with boundaries and those that are non-orientable.

Figure 3.3: Diagram of spanning tree.

Layered decomposition

The layered decomposition is a connectivity coding method introduced by Bajaj [Baj99] which deconstructs a triangle mesh into concentric vertex layers and constructs triangles within each of the pairs of the adjacent layers. The overall mesh connectivity is represented based upon three components; the total number of vertex layers, and the

layout of each vertex layer and of each triangle in each of the triangle layers. In an ideal case, the connectivity compression is reduced to coding the number of vertex layers, the number of vertices in each vertex layer, and the generalized triangle strip that exists in each triangle layer. However with the existence of branching points, bubble triangles and triangle fans, overhead bits are included. Branching points are created when a vertex layer self-intersects and it divides a vertex layer into segments called contours. This is illustrated in Figure 3.4.

1. **Generalized strip.** Vertices lie on two adjacent vertex layers
2. **Bubble triangles.** All vertices here belong to a single contour
3. **Triangle fan.** Vertices lie on two or three contours of a single vertex layer

Figure 3.4: Diagram of layered decomposition.

A similar layered decomposition which also dealt with the vertex span tree construction but differed in several areas:

- Vertex layers aren't combined into the vertex spanning tree.

- Its only accesses a small portion of vertices at each decompressing step.
- Applicable to any kind of mesh topology compared to manifold meshes.

Bajaj [Baj98] later adapted this method to extend to encode large CAD models to compress quadrilaterals and general polygonal models with smooth non-uniform rational B-splines (NURBS) patches.

Triangle conquest

The triangle conquest method is processed and unprocessed by inserting triangles by triangles into conquered regions of the mesh and outputs building operations into new triangles. The cut-border machine method was introduced by Gumhold and Straffer [Gum98] whose steps involve inserting new triangles into the conquered regions closed by the cut-border. This is represented from one of five possible operations: new vertex, forward, backward, split and closes. This sequence of building operations is then encoded by Huffman encoding. This algorithm can handle manifold meshes of both orientable and non-orientable, is notably fast in decompression and its decompression is capable of being implemented in parallel. Gumhold later improved the performance by applying an adaptive arithmetic coder and optimizing the border encoding [Gum99].

Rossignac [Ros99] developed edgebreaker algorithm which is similar to the cut-border machine method with the exception of its offset data associated with split operations are not encoded. Rather, edge loops control the triangle traversal bounding a conquered region. At each step of the algorithm the current focused edge loop is brought into what is

called the active gate while the others are stored for processing at a later point. Once a triangle is conquered it is assigned an operation code from the letters C, L, E, R, and S which is then processed into the final output and is illustrated in Figure 3.5. Assuming the removed triangle is enclosed by g , the active gate and v , the vertex, five possible output codes are:

- C (loop extension) if v is not on the edge loop
- L (left) if v immediately precedes g in the edge loop
- R (right) if v immediately follows g
- E (end) if v precedes and follows g
- S (split) otherwise

Figure 3.5: Example of the processed used in the Edgebreaker algorithm.

The compression process is done in a DFS traversal of the dual graph of the mesh. The split operation splits the current loop into two, sending one into the stack for later. The edgebreaker algorithm is a very versatile algorithm; it can encode topological data of orientable manifold meshes with multiple boundary loops or arbitrary genus. It is however unsuitable for streaming due to its required two pass decompression.

King and Rossignac [Baj99] improved the guaranteed worst-cased modified the edgebreaker algorithm. Syzimezak[Szy00] optimized methods for meshes with high

regularity.

3.1.2 Non-triangle based methods

In comparison to triangular meshes, not as much had been dedicated to non-triangular meshes consisting of arbitrary face degrees and of vertex valences. Although other degrees exist, the dominant element is the quadrilateral.

Early attempts to code general graphs which dealt with the connectivity component of a geometric mesh led to rates of approximately 9 BPV. These attempts are based upon constructing interlocking spanning trees for both vertices and faces. Chuang et al. [Chu98] described a compact code using canonical ordering and multiple parentheses. Li and Kuo [Li98] proposed a method that traverses the dual mesh among its edges and based on the edge type would output a variable length sequence of symbols which then coded using a context-based entropy coder.

Isenburg and Snoeyink introduced the FaceFixer [Ise00B] which codes the connectivity of polygon mesh with their properties. A mesh traversal is organized through successive gate labeling around the current active boundary loop. The way holes or faces are fixed while traversing an active gate are denoted by seven labels $\{F_n, R, L, S, E, H_n, Mkl\}$. This has been generalized to quadrilaterals, arbitrary polygons and hybrid triangle-quadrilateral meshes respectively by King et al [Kin99A], Konrod & Gotsman [Kro01] and Lee et al [Lee02].

3.1.3 Degree-valence approach

The generalization towards arbitrary polygon meshes was done after the valence coder of Touma-Gotsman [Tau98C] and relies on the concept of duality. This was developed independently by Khodakovsky et al [Kho02] and Isenburg [Ise02]. The key idea behind the degree-valence approach is that the compression should be self-dual; both the mesh and its respective dual will be coded with the same number of bits leading to two consequences:

1. Coding process should be symmetric in coding both the valences and degrees
2. Mesh bitrate should be measured in bits per edge since it is the only variability in a graph dualization.

This underlying concept is similar to most connectivity compression methods: the seed element is chosen until all of its neighbors are recursively traversed and until all connected components are processed. A symbol is outputted every time the encoder processes a new element. Algorithms differ in their traversal of the mesh and in the symbol sets used to identify the state of the encoder.

3.1.4 Geometry coding

In contrast to connectivity coding of which its performance is considered loss-less and very close at achieving an optimal compression rate, geometry coding is generally consider to be lossy and wasn't a significant area of research until further on after connectivity. Most single-rate geometry compression schemes involve three-step procedure:

1. Pre-quantization of vertex positions
2. Prediction of quantized positions
3. Entropy coding of predictive residuals

Quantization is a lossy procedure which takes advantage of human perception of precision of geometry data where typically each uncompressed coordinate component is an IEEE 32 bit floating number a quantized coordinate would be represented with a smaller set of values. Typically these types of schemes quantize each coordinate uniformly at 8-16 bit resolution. A global application of the same quantized resolution [Tau98B, Dee95, Baj99, Tau98C] and can also be portioned as in [Cho97] where different quantization resolution are adaptively chosen based on triangle size and local curvature.

3.1.5 Prediction

The prediction reduces the amount of geometry data and exploits the correlation between adjacent vertex coordinates. A good prediction scheme is classified as one which generates prediction errors with highly skewed distribution which then are entropy encoded. There have been several different geometry prediction schemes but all can be treated as special cases of linear [Tau98B] with specifically chosen coefficients.

Linear prediction was created by Taubin and Rossignac [Tau98B] where a vertex position is predicted based upon linear combinations of K previous vertices in the vertex spanning tree. These K previous vertex are uniquely selected along the path from the current to the

root vertex within the vertex spanning tree. This is represented in Equation 3.1 where the position v_n of the n^{th} vertex. The values $\lambda_{1,...,k}$ are chosen in order to minimize the mean square error.

$$v_n = \sum_{i=1}^K \lambda_i * v_{n-i} + \epsilon(v_n) \quad (3.1)$$

One of the more popular prediction schemes is the parallelogram prediction developed by Touma and Gotsman [Tou98]. A new vertex is encoded when a triangle with two vertices, u , v are on the active list. This scheme assumes the four vertices form a parallelogram and that the new vertex position can be predicted as $r_p^P = v + u - w$.

3.1.6 Vector Quantization

More recently vector quantization (VQ) has been proposed which differs from other standard geometric compression by first predicting vertex positions and then jointly compressing the three components of each predicted residual. Its main advantage over scalar quantization methods includes more freedom in cell shapes for quantization and better exploitation of dependence between vector components.

In Lee and Ko [Lee00], the Cartesian coordinates of a vertex are transformed into a model space vector using three previous vertex positions. The model space transform is regarded as a prediction and the model space vector as a prediction residual. The model space vector is then quantized using Lloyd's algorithm [Ger92].

Chow and Meng [Cho02] proposed a predictive vector quantization scheme which involved a simple predictor used to predict a new vertex from the midpoint of two previously traversed vertices to ensure a linear time complexity.

3.2 Shape Analysis

Shape analysis involves the analysis and processing of geometric shapes for shape recognition and similarity. Over the past decade, with advancements in modeling, digitizing and visualizing techniques of 3D data, there has been an expansion of increasing amounts of 3D models. Models have expanded in size and complexity to such a degree that the need for categorizing within mesh model databases such as the Princeton Shape Database [Ali98, Man98, Tau98C] University of Konstanz [All01], Utrecht University [Bar99, Van93] and Purdue University [Fol96, Han98] and experimental search engines. In order to both classify, store and retrieve, the need for better representation and analysis of 3D mesh models is needed. One method is the use of annotating models [Pru90] where a user creates textural annotations that help classify and describe an input 3D mesh model into a database. However this isn't practical method due to time consumption involved in creating annotations, not to mention its subjectivity based on user input and that it would fully represent the object in its purest form. Figure 3.6 shows an excerpt of the traditional framework used in 3D shape retrieval systems [Tan08].

Figure 3.6: Excerpt of 3D model database retrieval flow diagram.

3.2.1 Structure Detection

In Shikhare’s paper [Shi01] he presents a geometry compression method designed for engineering type 3D models, where he observed that traditional methods weren’t as efficient. This was because engineering models tend to have small components which lead to short graph traversal methods and these models usually have many non-smooth meshes which rendered predictive encoding and signal processing less suitable.

Shikhare’s method employs an automatic discovery of salient features within a given 3D model and uses it to reduce redundant occurrences of repeated features across the model in two levels; component level which compared full meshes against others and sub-components which analyze the vertices of a mesh based on a comparative BFS traversal from vertices which exhibited promising properties.

The paper by Pauly et al [Pau08] developed a framework to discover regularly repeating geometric structures in 3D shapes from both point and mesh models without any prior understanding of geometry or spatial location which could define the pattern. Their work discovers complex regular structure amongst clutter, noise and missing geometry. Their

method uses an analysis of pairwise similarity transformation that reveals a lattice structure in the transformation space.

In the paper by Langbern & Martin [Li06] their work differentiates from the concept of reverse engineering, noting how it generally approximates an object shape rather than a focus towards recording the underlying design intent. Their overall goal is to represent the geometric intent of a model which using feature trees augmented with geometric constraints describes regularities which they decompose into boundary representation models and into regularity feature trees. Their process describes an approach for finding regularity features which is based on recovering broken symmetries in the model that results in a tree of regularity feature for other subsequent use in regularity detection and selection.

3.2.2 Symmetry detection

The concept of symmetry is present in many computer graphic applications today including alignment, recognition, and segmentation; it is present in both natural objects, but more predominantly in man-made objects. In Podolak's paper [Pod06] they describe a planar reflective symmetry transform (PSRT) that can capture the continuous measure of reflection symmetry of a shape with respect to all possible planes. They use this PSRT to define two new geometric properties; the center of symmetry and the principal symmetry axis; this shows its usefulness in aligning objects in a canonical coordinate system. Their method provides an efficient Monte Carlo sampling algorithm to compute the transformation of surfaces and provide iterative refinement algorithm to find local

maxima in a precise manner.

In Mitra's paper [Mit06] they develop an algorithm that efficiently discovers a compact representation of Euclidean symmetry in one of three states; partial, approximate, or both. Their algorithm looks at local shape signatures in an appropriate transformation space and uses a clustering method to extract potential significant symmetries which are then verified.

Unlike in 2D, 3D symmetry detection is more complex. Jiang and Bunke [Jia94] developed a symmetry detection method which generates hypothetical axes of symmetry and verifies them to detect only rotational symmetry. Their method uses graph theory and works off a graph representation of the model. Brass and Knauer [Bra04] takes general 3D objects to test the congruence or symmetry which is capable of retrieving symmetry groups of arbitrary shapes. Minovic [Min93] worked with an octree representation and uses an octree traversal to identify symmetry. In order to find the candidate axis they use PCA. These methods were dependent on mesh topology and are sensitive to both noise and data imperfections.

Later on Kazhdan [Kaz04] introduced symmetry detectors which were spherical function descriptors of a model's measure of rotational and symmetry with respect to each axis through its center of mass. However, their algorithm was very costly due to the required accurate sampling in all possible directions. Even later Martinet [Mar06] developed a method that deterministically finds generalized moments which examines the extrema

and spherical harmonic coefficients of these moments to recover symmetric parameters. They use an incremental algorithm of the whole shape based upon symmetries of its subparts.

3.2.3 Reverse engineering

The concept of being able to approximate 3D objects has been an intriguing area for quite some time, especially being able to develop parametric parts used to describe them. These parametric descriptors allow the ability to create at different levels of abstraction while being able to retain important information. However, most digital media lacks semantic descriptions and reverse engineering a model's structure and regularity from its 3D geometry is a difficult problem [Sin98].

A large portion of reverse engineering 3D mesh models is based upon creating a collection of curves or a curve network. The idea behind it is that in models, especially those that are non-organic, can be represented by a sparse collection of curves describing the key characteristics of a given 3D mesh model. One of the earlier papers behind this included Wires [Sin98] which introduced a novel modeling paradigm for space deformations. Their work created curve networks from traditional CAD/CAM using free-form surface patches.

In [Duc03] they specifically targeted 3D NURBS models and designed a system with guaranteed distortion that was based upon an entropy encoded differential pulse (DCPM). Their system worked upon the underlying model in its synthesis stage to be used for

efficient coding without complex analysis. Along with this work, their research also examined NURBS parameters including knots and researched the relation between knot and surface distortion.

In [Duc03] a low-level analysis is performed which extracts a model's feature curves. This provides a compact vector representation at a desired abstraction level that bypasses the difficult task of reverse engineering the detection of global structures while preserving a model's main characteristic feature.

More recently the use of curve-networks has been combined with sketch-based modeling systems as in iWires [Gal09] which infer the shape based on user input and sketches. While addressing their interpretation from 2D input curves, the inclusion of a user input helps relieve many of the topological and geometric restrictions that are normally a result from a classic curve network. They also combine the idea of adding additional descriptive information about their shape relative to others within the same network.

Conversely, [Lit01] targets subdivision 3D models, particularly mechanical and CAD models. They present a framework which segments control polyhedron polygon meshes into near constant curve surface patches which approximate boundaries for each patch which are then assembled and encoded. Although they share the general goal of compressing information derived from subdivision models, they approach mesh decomposition from a geometric analysis perspective and ignore the input mesh topology whereas we approach it heavily upon the topology and connectivity information from the

initial mesh.

3.2.4 Inverse procedural modeling

Unlike procedural modeling methods which use a set of inputs, most commonly grammar based rules or textures such as those used to create plants [Pru90], buildings [Won03] and cities [Par01], inverse procedural modeling is the reverse of this; it is an explicit set of procedural rules that are constructed to encode the necessary instructions to efficiently build objects. Rules (e.g. shape grammars) are derived from example geometry. This is beyond simple geometry synthesis which creates similar models from examples that describe the structure of the model and of its space [Mer07, Lag05].

One of the earliest approaches was developed by Hart and Flynn [Coc97] which used similarity hashing to create linear fractions. With this, they developed fractal branching rules for L-systems based off of the concept of geometric hashing in 2D graphics.

Other work revolved around examples to help derive geometric related synthesis of context based [Sha04], geometry [Lag05] and models [Mer07]. Kravoey [Kra08] developed non-homogenous resizing of complex models. They used regular pattern detection methods that were based upon example geometry which lead to discovering structural regularities.

This method was later developed into variations that were based off of exemplars by changing its replication frequency or by simultaneously symmetric piece editing.

Whereas Brokeleh [Bok10] used partial symmetry and pairwise matching to help detect and modify new structures based on a common set of input geometry.

In interactive approaches, iWires [Gal09] developed an interactive approach to modifying a given object based on a curve network that analyzed its underlying structure to maintain its regularity. Aliaga [Ali07] which developed style grammars for interactive visualization of architecture. St'ava [Sta10] which investigated the automatic generation approach towards the problem of inverse procedural modeling.

A large body of work still appears to focus on triangulated mesh models [Shi01], and only a limited work has targeted subdivision models. Model analysis is largely based upon geometric methods of analysis [Gal09, Meh09]. In terms of model recreation, most methods appear to be limiting in how it reconstructs the re-created surface, namely that the original topology is ignored over its geometric properties [Gal09, Meh09] when re-creating the surface. Our work addresses these prevalent issues in the areas of 3D modeling and reconstruction research. By focusing on subdivision polygon models, our work strives to maintain the inherit topology found in the original source models using a graph-based heuristic that has significant compression capabilities.

Chapter 4

Method

In this chapter we describe in detail the algorithms used to take a subdivided 3D polygon model as input and deconstruct the necessary unique elements that reflect the intrinsic underlying topology and then to be able to separately recreate the original mesh from a unique file description.

This method distinguishes itself from feature-element discovery methods that are based upon geometric analysis described in [Shi01], rather than CAD-based reconstruction methodology seen in [Gal09,]. Those curves are based upon geometric properties, usually the normal differences.

The overall implementation is outlined in the chart diagram in Figure 4.1. The process is split up between the main algorithm (represented in heavy lines) and optional methods (represented by dashed lines) that take a given input of a 3D subdivision mesh polygon model (OBJ format) which is compressed into an MPD

file format. This is then decompressed and re-created back into an OBJ file format.

The compression component of the process is composed of the main algorithm which consists of macro decomposition, micro decomposition and the writing to file stage with the option to be further compressed during the process in the approximation control and spatial analysis stage.

The given input is first processed through the macro decomposition as described in Section 4.1 which identifies the unique mesh from instances (copies of the unique mesh). From here the unique meshes are then passed along in the micro decomposition stage in Section 4.2 where they are further analyzed based upon the underlying graph topological structure. At this point all the instance information of both the meshes and surface patches identified can be analyzed optionally for any possible spatial relations that exist. Finally all of the information from both decomposition and the optional spatial and approximation control is then processed into our proprietary custom file format. The file is then re-constructed into an OBJ file format again which is covered in Section 4.4.

Figure 4.1: Visual diagram of model reconstruction algorithm.

4.1 Macro Decomposition

At this stage we take apart the given input subdivision polygon mesh model into separate meshes, identifying the unique (first) mesh from any subsequent copies. In this section we work extensively off the component-level algorithm (mesh comparison) described in Shikhare's work [Shi01]. Unique meshes are explicitly analyzed and represented in the subsequent steps leading to file storing. Instance meshes (copies) of the respective unique meshes are stored based on their geometric position relative to their master (unique) mesh and further analyzed in the spatial stage. We describe below the main steps in both identifying and storing the master and instance meshes.

The process begins first with separating all the meshes within the model. For every identified mesh we then normalize its pose that is to normalize its translation, scale and rotation. The translation is normalized based off of the mesh's centroid and, unlike in Shikahre's work we also normalize its scale. The rotation is then normalized based upon its PCA (Principal Component Analysis).

At this point we perform a one-on-one comparison from the first mesh (the candidate unique mesh) to the second (possible instance mesh) based on their local positioning in order to determine whether the two match. Another difference compared to Shikhare's method here is that the one-on-one comparison

is done on a BFS connected component list which performs its verification analysis based on its neighborhood connectivity. This allows for another verification flag within the verification stage to determine if it's an unsuitable match before finishing the list of vertices.

Once a mesh has been identified as a copy of another it is stored based on its transformation vector from the original's centroid which consists of its translation vector and rotation vector. The rotation is stored as 3 vectors based off the rotation on the X, Y, Z axis. This results in a list of instance meshes which are copies of the first unique mesh; however we discuss an alternative to listing every instance mesh position and improve the storage of this instance mesh information in Section 4.6.

4.2 Micro Decomposition

For each of the unique meshes that have been separated from the instance geometry, the mesh is then decomposed into surface patches. This is implicitly related to the underlying topology generated from the input mesh. Figure 4.2 illustrates the key stages of the micro decomposition of a given unique mesh. First extraordinary vertices are identified which is further described in Section 4.2.1. This is then followed by heuristically growing linear curves which emanate from those identified as extraordinary vertices which are further described in detail in Section 4.2.2. These curves are then arranged to make up the surface patches

which make up the input unique mesh and is further explained in Section 4.2.3.

Figure 4.2: Breakdown of micro decomposition algorithm.

4.2.1 Extraordinary Vertices

Our work centers around the notion of the mesh topology, specifically how the topology among a mesh surface changes around its extraordinary vertices as features are added or removed. As we explore this concept we differentiate extraordinary vertices into two types, based on their valence (vertex degree) with respect to the common vertex valence of a subdivision model (four), either being higher or lower. This is easily done upon checking the vertex degree when reading from the original source file.

4.2.2 Curves

From here, curves are stored based on the edge loops which emanate from the

extraordinary vertices found among the mesh surface. It is important to note that the method described in this section can be easily adapted to the winged-edge data structure for easy edge navigation.

The curves are found based on a parallel growth on a given model expanding from all edges incident to the extraordinary vertices. The process of the overall curve creation is achieved in two phases: the growth and termination phase and is illustrated in Figure 4.5.

The growth phase begins with each edge incident to the set of extraordinary vertices as seen in step 'a' and 'b' as shown in Figure 4.3. From this point until the curve's termination, each curve undergoes an expansion of one edge from their respective edge loops at a time (steps b – e). There are two ways a curve can terminate: either due to a merge, or when a split occurs. A merge event occurs when the current curve's end point matches that of another curve. The current curve would then inherit all of the other curve information and the other curve would be removed from the current list. A split event occurs when one curve intersects another curve that's end point has already passed that point along the edge loop. For the purpose of determining the surface patches in Section 4.2.3 the curve ID of the current curve is stored on the other curve and vice versa.

Figure 4.3: Curve component breakdown and the various functions involved in the micro decomposition phase.

Upon analyzing given subdivision polygon models, we've noticed how their boundary curves are modified based on how they are grown in sequence. For example taking a simple action of extruding a face on a planar sheet as illustrated in Figure 4.4. The resultant action on its topology creates two distinct sets of extraordinary vertices; lower vertices on the original faces extruded identified by small circles and higher on the bounded faces surrounding the extruded face identified by squares.

Figure 4.4: Occurrence of extraordinary vertices resulting from extrusion.

Given this observation we examined three different methods of growing the curves:

- Lower extraordinary vertices first.
- Higher extraordinary vertices first.
- No priority weight given.

The first two methods place priority on either set of extraordinary vertices such that those curves emanated grow until termination at which point the other set will

begin and proceed until all edge terminate as well. The third method gives no priority of the weighting of the extraordinary vertices and all edge loops grow simultaneously one edge at a time together until all curves terminate.

The lower extraordinary vertices appeared to perform well in objects with relatively low complexity (a model that is close to that of a primitive object). Yet with the 3D subdivision models tested it appears that the first two methods generally produced long slivers of surface patches; these are long patches with one axis consisting of only one polygon in length. The issue will be that if we were to introduce a form of compacting adjacent patches of similar dimensions or shapes either of the first two methods could still prove more efficient at growing.

As we already stated, the overall topology is sensitive to any changes to the mesh. What was once a simple mesh topology could have an increased complexity based upon any additional feature, especially if it is done incorrectly. To better illustrate this issue we examine another common element in subdivision 3D modeling, cylinders or circular patches. The reason behind this is that there are several unique ways such a circular patch can be filled and as such could affect the overall topological structure. Looking primarily at the caps, at either end of the cylinder primitive, we can observe the different methods for filling the circular boundaries with only quadrilaterals. Depending on the method we can see how drastically the resultant topological structure can change. Table 3 is a breakdown of possible methods of filling a circular boundary based on the number of

enclosed edges that make up the boundary, the resultant quadrilaterals required and the resultant topological structure of a cylinder.

Table 3: Polygon cylinder caps and their resultant underlying topological structure.

Polygon Degree	Parallel	Radial	Square Patches
8
12
16

4.2.3 Surface Patches

Similarly to the orientation of a polygon faces at the moment we have only experimented using simple patches which consist of four curves and radial patches. Theoretically there would exist three possible patch configurations, illustrated in Figure 4.6.

- **Simple patches.** These would consist of 4 curves illustrated in Figure 4.5.
- **Complex patches.** These would consist of more than four curves.
- **Radial patches.** These would also consist of more or less than four curves denoted by an 'R' in the file output.

Figure 4.5: Simple patch description.

Figure 4.5 describes the breakdown of a simple patch, consisting of 4 linear

curves which meet at four points A,B,C, D where each opposing pair of curves would have the same number of vertices. The complex patch would be composed of 2 or more simple patches designated by the vertex or vertices within the boundary that have a degree higher than three. A radial patch is unlike the other 2 patch types and accounts for circular regions which are enclosed by multiple curves which are equidistant from the same extraordinary vertex source.

Figure 4.6: Patch type classification for surface patches.

Those curves which haven't identified their adjacent neighbor curves are first stored in a queue and their method of finding the rightmost adjacent curve is illustrated in Figure 4.7. For each curve we identify both directions, going from A to B and vice versa, based upon the surface direction its front right (B to FR) and back right (A to BR) respectively. The operation to find the rightmost in any one direction is done in the following method. For each curve in the queue, we locate the last two points (b and B) and determine the rightmost edge that connects to it (fr), the local front right edge. With this front right edge (B, fr) we then locate the curve that contains points (B, fr) in sequence and determine the opposing point

that lies on the other side of the curve (FR). The opposite direction operates on a similar method but locates the first 2 points instead (a and A).

Figure 4.7: Diagram of the right turn concept based upon the surface normal direction.

One major problem we've noticed with this method is when geometric surface changes dramatically inside the patch description. Generally with flat surfaces the interpolation with Coons [Coo67] would be able to handle if not automatically clean the surface topology when filling the patch, however in cases where the geometric surfaces change its flow it's geometry is not entirely captured accurately because of its opposing boundary curve along its axis has a different geometric layout. Figure 4.11 illustrates this issue more clearly and Section 4.5 covers a theoretical method of resolving this issue.

4.3 File Format

In this section we describe the format of the MPD (mobile procedural description) file description which stores all the relevant information that is to be recreated later in Section 4.4. The format of writing this information and storing such information is modeled after OBJ file format due to the simplicity of information kept in order as well as to focus on essentials instead of having verbose information stored. For an example of an MPD file is explained in Appendix A.

The MPD file would contain the following elements in order as it would appear in a given MPD file.

- Meshes
- Vertices
- Curves (list of vertex IDs from vertices list above)
- Actions (if applicable)
- Patches (list of curve IDs from the curve list above)

Meshes

This section of the file follows suit from Shikhare's output structure. A list of instances is first stored with 3 vectors, offset translation vector, 3 rotation axis, and one final vector to represent scale. The following information would describe the master mesh.

```
M {translation vector} {rotation axis} {scale
vector}
```

(3.1)

Vertices

This would write out all unique vertices stored in the curves as described in Sections 4.2.1, 4.2.2 and, if applicable additional curves in Section 4.5. This is described in the same form as described in OBJ format with the flag “V” followed by the 3D coordinates from the model (in world space).

```
V { xyz coordinate}
V 0.4 0.50 0.44
```

(3.2)

Curves

For each curve identifies the ID referring to the vertex component that make up the curve, this is similar to the face-vertex and vertex relationship in OBJ file format.

```
C {vertex IDs}
C 0 2 4 5 3
```

(3.3)

Patches

Each patch follows the same relation as with curves, that is each patch is composed of curve identifiers of the curve descriptions as previously mentioned. To accommodate the direction of each curve, the index for the curves begins at 1

and any ID that is negated represents a curve direction in the opposite direction and would be reversed in the reconstruction phase.

```
P {curve IDs}
P 1 -4 2 3
```

(3.4)

Due to the different classification of patches, we assign an additional flag after P for any patch type that is not a simple patch C and R for complex and radial respectively. Any captured instances are described immediately below the current patch flagged with an I .

Actions

To further improve the data stored in we also describe two forms of spatial relationships to any instance of either the meshes or patches: ROT and LIN for a rotational relationship and a linear step relationship. This spatial analysis is further described in Section 4.6.

```
ROT {#copies} {deg} {axis: x y z} {origin: x y z}
LIN {#copies} { translation vector}
```

(3.5)
(3.6)

4.4 Re-creation

At this point we describe the method used to recreate each of the surface patches

identified for each of the given unique meshes. The order of recreation is represented by the order as described in the MPD file from the previous section. The process of re-creation for every unique mesh or unique surface patch is as follows. First every instance transformation is temporarily stored, followed by recreating the master copy, which is then separately stored as well. This master is then duplicated for each instance transformation stored and applying this respective local transformation.

When re-creating a surface patch the re-creation process first determines the patch type and whether it is simple, complex or radial. This is then processed (if it isn't a simple patch type), to determine the size of the patch. If it is a radial patch; it determines the number of surrounding curves. Especially in a radial patch type, the number and the parity determine the type of patch filling method is illustrated briefly in Figure 4.8. For surface patches that have the common size of 4 we proceed with the Coon's interpolation algorithm which is explained in Section 4.4.1.

Figure 4.8: Patch filling reference chart.

4.4.1 Coons interpolation

Due to the nature of the subject matter being mechanical, the most appropriate choice was to implement Coon's interpolation algorithm to recreate the surface patches. Especially dealing with non-organic models, their surface is best represented as quadrilaterals of proportional size and set approximately equidistant from one another. Figure 4.9 illustrates Coon's algorithm which is presented in Equation 3.7.

Figure 4.9: Diagram of Coon's algorithm.

$$\begin{aligned} P[i, j] = & u_2(1 - v) + u_1(v) \\ & + v_1(1 - u) + v_2(u) \\ & - (A(1 - u)(1 - v) + Du(1 - v) + B(1 - u)(v) + Cuv) \end{aligned} \quad (3.7)$$

4.5 Approximation Control

Although not as extensive as originally envisaged, our method breaks this process

into two additional options from the general method described in this chapter. This provides two varying levels of condensing such information and one to include further resolution. Figure 4.10 illustrates a simplified patch with the grey lines representing the resultant edges and the black lines as curves which are stored.

Figure 4.10: Diagram of approximation control on a simple patch.

The reduction method includes two phases of for condensing information further. This process is preformed patch by patch of the unique meshes in a given 3D model. The reduction components work by removing non-essential edges to the patch description. When ensuring the topology remains consistent we first check any selected edges up for removal to see if they are crucial in maintaining the topology. This is done by verifying whether they emanate from any of extraordinary vertices among the mesh surface. A simple way to check for this is by providing a Boolean flag which monitors if it emanates from an extraordinary vertex. Otherwise if the flag is false it is an acceptable for removable without affecting the overall topology.

The first concerns itself with any relevant bevel information while the second begins a collapsing process to a desired level. Traditionally in common polygon subdivision modeling, applying bevel details around specific edges is something done in order to add further control over hard edges. Here we present the method which examines any edge loops within the given patch that are within a close proximity to another patch.

The second level of patch reduction at this stage offers a desired level of reduction and will collapse an N set of non-essential edges along each of the edges. This adds the benefit of reducing the model to only essential edges which makes up the underling topology of the original mesh and retaining the details.

Extra information is stored to obtain better surface representation. At this point the only available solution obtained is taking the approach other methods acquire but for differing purposes. Unlike other papers which use geometric properties to fully represent a model, we examine the geometric properties for a significant displacement in the geometric surface's representation on a given patch. We look at the surface normal along both axes on the patch representation to notice any sufficient shift in curvature lines. Figure 4.11 illustrates below the unwanted result of some patch recreation and the proposed solution of such issue by storing additional curves represented by dark lines.

Figure 4.11: Example issue with reconstruction and proposed solution with additional curve information stored based on geometric change along a patch axis.

4.6 Spatial Analysis

It is generally accepted that using simple geometric relationships can further reduce the representational information of mesh instances. This is especially true when considering that, when modeling, it is common to build a set of polygons or even full meshes then apply discrete transformations. Examples of this can be seen in the following Figure 4.12.

Figure 4.12: Examples of discrete transformations seen in meshes (a and b) and surface patches (c and d) of linear type (a and c) and rotational (b and d)

Among other geometric relationships (including symmetry) we choose to focus on straight line segments and circular sweeps based on several reasons:

- They are among the simplest geometric relationships to describe
- Their cost of recording versus data saved is the most efficient
- It is among the most common actions used in creating duplicated meshes in a given model.

The process of determining such a spatial relationship begins with all instances stored for a given master mesh or for a given master patch. Each instance is represented by its position (centroid). This process begins with a pre-initialized phase which first initializes the distance from every position and to every other one in the set. A queue is assigned for each position and contains the distances to every other position in ascending order.

The first step of the process begins by selecting a source point (step A) which is any point within the search query list of positions currently established. The second step begins by selecting the point with the shortest distance (step A). This is the second point in the current candidate spatial pattern we store both its position and distance. From this second point we look at its neighbors (steps B and C) to determine if any of the neighboring distances are equidistant (within a margin of error to the source point). If there is a match we proceed into a third step. If any neighbor points have been found we go into the final fourth step. Otherwise if the current point has no detected pattern found continue searching through the next point with the next shortest known distance to the current source point.

The fourth step begins by first identifying the pattern type; whether the current three points identified thus far that either form a circular (step E) or linear relationship (step I). A linear relationship is checked first; determined if the three points are collinear. If not, then the points are checked for a circular relationship;

this is determined by verifying against the origin of the circular relationship. The origin is calculated based upon the intersection between the two perpendicular bisectors of the two chords of the first three points (step B).

From here onward we continue looking for additional points that match the same pattern identified at the beginning of this step until either the source pattern is identified as a neighbor again (closed circle) or no neighboring points are found that match the same distance as established in the first step (step F). Once terminated we store all the points established in the pattern thus far as well the specific spatial pattern established (step G). All of these identified points are then removed from the search query of points (identified as black points in Figure 4.13).

Figure 4.13: Visual storyboard of spatial analysis process.

Once if any relationships are determined, the information stored is described as follows. Circular information is based on an axis vector, origin of circular sweep and the number of instances from the original. Circle information is determined based on any two chords between any 3 instances. The origin of this rotational axis is found by determining the point of intersection between the perpendicular bisectors of each chord. The amount of rotation (degrees) is determined based upon any two consecutive points and the origin. The line segment is stored based on the slope (3 dimensional vector), the number of iterations from the origin instance. The methods of storing the required information are illustrated in Figure 4.14.

Figure 4.14: Diagram of shape analysis construction process for the rotational and linear occurrences.

Chapter 5

Results

Our method was tested on a range of models of various resolutions representative of game and film modeling practices. The models in Chair 02 and Chair 03 were acquired from TurboSquid.com, an online repository of read-made 3D models and assets for film and game production. These models were also multi-part models.

In Sections 5.1 and 5.2 we showcase our method’s results in areas of compression and topological abstraction, in addition to demonstrating its usefulness as a topological analysis tool.

5.1 Micro Results

5.1.1 File Compression

Table 4: File Compression Results

Model	Original (OBJ)			Recreated (MPD)			Reduction (MPD/OBJ)		
	#Vertices	#Edges	File Size (KB)	#Vertices	#Edges	File Size (KB)	#Vertices (%)	#Edges (%)	File Size (%)
Bottle cap	1634	3264	412.0	174	156	8.05	10.65	4.78	1.95
Bottle	1794	3584	289.0	863	252	45.80	48.10	7.03	15.85
Can	484	964	73.8	179	180	9.00	36.98	18.67	12.20
Chair02	3339	6704	594.0	474	405	19.60	14.20	6.04	3.30
Chair03	18218	36400	3832.0	1045	1124	51.90	5.74	3.09	1.35
Misc02	376	752	61.3	145	153	6.29	38.56	20.35	10.26
Misc03	769	1537	130.0	225	234	26.80	29.26	15.22	8.46
Hydrant	3662	7320	633.0	450	574	26.80	12.29	7.84	4.23

5.1.2 Underlying topology structure

The darkened lines represent the topological structure consisting of curves generated by the algorithm in the micro decomposition (Section 4.3). The red

generated by the algorithm in the micro decomposition (Section 4.3) .The red patches represent patches with are flagged by patches which are further reduced based on their spatial relation to other relative patches.

The following diagrams, Figure 5.1 -5.8, illustrate the underlying topology structure. For each model presented we present the following states:

- a. **Original model.** Including its wireframe for visual references
- b. **Captured topology.** The darkened lines represent the topological structure composed of captured curves.
- c. **Captured topology.** Same as captured topology but in XRAY view to showcase the overall structure more clearly.
- d. **Re-creation model.** The model recreated using with our method.
- e. **Unique patches.** A visual reference for any unique patches which benefit from a shape analysis compression. It should be noted this doesn't account for every unique patch located on the mesh.

We also describe the total number of curves and patches recorded in the MPD file of the original model.

Figure 5.1: Topological analysis for model (Bottle cap).

Figure 5.2: Topological analysis for model (Bottle).

Figure 5.3: Topological analysis for model (Can).

a

b

c

#Curves: 50

#Patches: 32

d

e

Figure 5.4: Topological analysis for model (Misc. Shape 02).

Figure 5.5: Topological analysis for model (Misc. Shape 03).

Figure 5.6: Topological analysis for model (Chair 02).

Figure 5.7: Topological analysis for model (Chair 03).

Figure 5.8: Topological analysis for model (Fire hydrant).

5.2 Recreation

In this section we show a visual representation illustrating the magnitude of reconstruction error per vertex in the mesh. The associated gradient ramp illustrates the magnitude of vertex error based upon the Euclidean distance from each recreated vertex to the nearest point on the original surface. The worst case vertex error is called the Hausdorff distance (Equation 2.13), and has been frequently used to compare similarities between two 3D meshes [Nut11]. We follow the traditional convention of expressing the distance as a percentage of the model's bounding box diagonal length.

In addition to the worst case scenario, we also take note of the mean error across the entire surface, again normalized by the diagonal. Numerical results are expressed in Table 5, while Figures 5.11 - 5.16 show a visual representation of the normalized Hausdorff values for each vertex on each of the models. In those figures, the leftmost column illustrates the original input model while the rightmost column illustrates the recreated results where the placement of red indicates errors of the vertices and the intensity of red indicates the error intensity.

Table 5: Recreation results

Model ID	Bounding Box Diagonal Length				Hausdorff Distance	Normalised Hausdorff Distance (% bounding box Diagonal)	Mean Hausdorff Overall	Normalised Hausdorff Mean (% bounding box Diagonal)
	X	Y	Z	Diagonal				
Bottle cap	23.564	3.320	23.564	33.489	0.215	0.64%	0.0218	0.07%
Bottle	21.123	63.089	21.150	69.812	0.476	0.68%	0.0397	0.06%
Can	5.619	11.254	5.570	13.757	0.218	1.59%	0.0310	0.23%
Chair 02	20.832	48.77	20.832	56.98	0.411	0.72%	0.0329	0.06%
Chair 03	22.204	9.329	9.484	25.884	0.173	0.67%	0.0524	0.20%
Misc. Shape 02	12.000	23.314	23.410	35.151	0.601	1.71%	0.0685	0.19%
Misc. Shape 03	61.288	50.211	16.900	81.012	0.952	1.17%	0.4966	0.61%
Fire Hydrant	16.912	30.423	14.447	37.687	0.478	1.27%	0.0356	0.09%

Figure 5.9: Bottle cap model, both original and recreated result.

Figure 5.10: Bottle model, both original and recreated result.

Figure 5.11: Can model, both original and recreated result.

Figure 5.12: Chair 02 model, both original and recreated result.

Figure 5.13: Chair 03 model, both original and recreated result.

Figure 5.14: Misc. shape 02 model, both original and recreated result.

Figure 5.15: Misc. shape 03 model, both original and recreated result.

Figure 5.16: Fire hydrant model, both original and recreated result.

5.3 Observation and Results

The results of our micro decomposition successfully found the underlying topological structure for a variety of models.

While the visual results are hard to distinguish between the original and the reconstructed, the compression values were also very solid. Given that our method is not loss-less and is limited to the graph representation, we still managed to capture and recreate the model with an average of 99.5% geometric similarity to the original model (value based on the last result column in Table 5). The Hausdorff distance also showed that on average the largest error was less than 2% of the model's bounding box diagonal.

On its own, our method produced impressive results as a lossy compression method. It relies on energy minimization techniques which can be very costly when applied to all possible pairs of matching meshes. By using our topological abstraction and its leaner set of information, we can accelerate Shikhare and similar algorithms for usage in film and game modeling (or just destructive subdivision modeling as a whole).

One of the known drawbacks of our recreation process is the problematic results of the Coons algorithm used on round surfaces. This can be specifically observed in our miscellaneous shapes (Misc. Shape 02 and Misc. Shape 03), which were designed to test the limits of how our algorithm handles complex topology including extreme shifts in

curvature on low poly models. Even in these areas of high degrees of curvature this only resulted in errors under 2%.

Another major drawback apparent is its inability to handle patches with major shifts in its geometric surfaces. This can be seen in the middle features of the fire hydrant model as well as the prominent error in Misc. Shape 02 along the curved ring. However, we still believe that introducing only a few key additional curves as described in Section 4.3 along these patches would greatly improve the reconstruction result.

A favorable side-effect noticed in some of our models is the ability to clean up topology in some areas, prominently seen in areas that captured the surface patch quite well such as the bottom of the bottle model. Despite reading as errors, this could be interpreted as a false positive to some as providing a smooth surface and fixing any areas of unwanted noise. It should also be noted that this is not currently ideal in all situations as seen in the Misc. Shape 03 where the curved top that turns into a square tip is averaged out over the course of the patch.

Although our work has focused on non-organic objects we have briefly looked into the adaptability of our work in this regard. The following figure is an example of the graph topological structure of a given face model. We felt that the face would be a prime example of sufficient feature complexity. The process looks promising however its felt that similarly to non-organic, organic micro decomposition would benefit from a post-process which optimizes the patches into a series of complex patches as described in

Section 4.2. Furthermore unlike in non-organic, organic models tend to contain some level of noise in its geometry which isn't accounted for in this method. As described in Chapter 2, the intention of organic topology may have additional factors that can be taken advantage of such as how the topology should reflect the underlying musculature and anatomy.

Chapter 6

Conclusion

6.1 Conclusions

Our decision to focus solely on the graph representation while ignoring any geometric information proved challenging yet the most fruitful. We succeeded in developing a graph-based heuristical analysis of the model's most essential visual and topological qualities, without sacrificing general accuracy in the area of geometric compression. Our method is quick and robust to changes in geometry. The insights gained from limiting our interaction with the model to its graphical essence provided the opportunity to build what we see and is a foundation and starting point for more sophisticated and specialized analysis methods. Results show impressive compression rates compared with state of the art algorithms.

6.2 Future Research Directions

As far as we know, the concept discussed here is still generally unexplored. The following list outlines several areas suggested as possible directions for future research topics.

- **Improve geometry based re-creation precision.** Building on top of our graph-based scaffolding, costly geometric information can now be selectively developed for specific needs. For example, one might improve the reconstruction algorithm to account for the loss of the curvature present in the source model. This would involve a more elaborate algorithm that would theoretically combine Coon's patch algorithm with a hierarchical extrusion procedure which can be sampled at different resolutions of accuracy (similar to level of detail progression).
- **Revise patch decomposition.** Another major component would be to further revise the patch deconstruction. One direction would be to address the complexities in large subdivision models as the intricacies of its topologies can become quite complex. At this stage it is suspected that the benefits of these concepts could be realistically applied to the 3D modeling industry. The other direction would be to further broaden the scope and accept triangulated models. One benefit of subdivision models is the enforcement of using quadrilateral polygons which allow the ability of edge and face loops. If this process could find appropriate similarities among triangle meshes where loops are either discovered or interpreted then this could apply to a wider range of applications and industries.

References

- [Abd10] H.Abd, L.J.Williams. Principal Component Analysis. Wiley Interdisciplinary Reviews: Computational Statistics, 2, (2010).
- [Ali07] D.G. Aliaga , P.A. Rosen, D. R. Berkins "Style Grammers for Interactive Visualization of Architecture." IEEE Transactions on Visualization and Computer Vol. 13 No. 4, pp. 786-97, (2007)
- [Ali99] D. Aliaga, J. Cohen, A. Wilson, E. Baker, H. Zhang, C. Erikson, K. Hoff, T. Hudson, W. Stuerzlinger, R. Bastos, M. Whitton, F. Brooks, and D. Manocha. "MMR:An integrated massive model rendering system using geometric and image-based acceleration." In Proceedings of ACM Symposium on Interactive 3D Graphics (1999)
- [All01] P. Alliez and M. Desbrun. "Valence-driven connectivity encoding of 3D meshes". In Computer Graphics Forum, Vol. 20, No. 3, pp. 480-489, (2001)
- [All03] P. Alliez and C.Gotsman. "Recent advances in compression of 3D meshes". In Advances in Multiresolution for Geometric Modeling (2003)
- [Ark96] E. M. Arkin, M. Held, J. S. B. Mitchel, S. Skiena, "Hamiltonian triangulations for fast rendering." Visualization Computing Vol. 12, No. 9, pp. 429, (1996)
- [Asp02] N.Aspert, D.Santa-Cruz, T.Ebrahimi, "MESH: measuring errors between surfaces using the Hausdorff distance," Multimedia and Expo, Proceedings. 2002 IEEE International Conference on, Vol. 1, pp. 705-708, (2002)
- [Att06] M. Attene, B. Falcidieno, M. Pagnuolo. "Hierarchical mesh segmentation based on fitting primitives." Visual Vol. 22, No. 3, pp.181-193, (2006)

- [Att07] M. Attene, F. Robbiano, M. Spagnuolo, and B. Falcidieno. "Semantic annotation of 3D surface meshes based on feature characterization". In Proceedings of the semantic and digital media technologies 2nd international conference on Semantic Multimedia (SAMT'07). Springer-Verlag, Berlin, Heidelberg, pp.126-139, (2007)
- [Baj98] C. Bajaj, V. Pascucci, and G. Zhuang, "Compression and coding of large cad models". Technical Report, University of Texas, (1998)
- [Baj99] C. L. Bajaj, V. Pascucci, and G. Zhuang, "Single resolution compression of arbitrary triangular meshes with properties." Computer Geometry Theory, Applications Vol. 14 pp. 167-186, (1999)
- [Bar99] G. Barequet, C. Duncan, and S. Kumar. "RSVP: A geometric toolkit for controlled repair of solid models." IEEE Transactions on Visualization and Computer Graphics, Vol. 4, No. 2, (1999)
- [Ben01] P. Benko, R. Martin, and T. Várady. "Algorithms for reverse engineering boundary representation models." In Computer-Aided Design, 33 pp. 839-851, (2001)
- [Ber05] R. Berndt, D. W. Fellner, and S. Havemann. "Generative 3D models: a key to more information within less bandwidth at higher quality." In Proceedings of the tenth international conference on 3D Web technology (Web3D '05). ACM, pp. 111-121, (2005)
- [Bok10] M. Bokeloh, M. Wand, and H.-Peter Seidel. "A connection between partial symmetry and inverse procedural modeling." ACM Transactions on Graphics. Vol. 29, No. 4, Article 104, (2010)
- [Bra04] P. Brass and C. Knauer. "Testing congruence and symmetry for general 3-dimensional objects." Computer Geometry: Theory and Applications. Vol. 27, No. 1, (2004)
- [Cho02] P. H. Chow, and T. H. Meng. "Vertex data compression through vector quantization." IEEE Transactions on Visualization and Computer Graphics, Vol. 8, No. 4, pp. 373-382, (2002)

- [Cho97] M. Chow, "Optimized geometry compression for real-time rendering." In IEEE Visualization, pp. 347-354, (1997)
- [Chu98] R.C-N. Chuang, A. Garg, X. He, M-Y. Kao, H-I Lu. "Compact Encodings of Planar Graphs via Canonical Orderings and Multiple Parentheses. ICALP: Annual International Colloquium on Automata, Languages and programming, pages 118-129, (1998)
- [Coc97] W. O. Cochran and J. C. Hart and P. J. Flynn. "Similarity hashing: A computer vision solution to the inverse problem of linear fractal." (1997).
- [Coo67] S. A. Coons. "Surfaces for Computer-aided design of space forms." Technical Report MAC-TR-41, Project MAC, MIT, (1967)
- [De98] T. DeRose, M. Kass, T. Truong, "Subdivision surfaces in character animation", Proceedings of SIGGRAPH (1998)
- [Dee95] G. M. Deering, "Geometry Compression." In ACM SIGGRAPH, pp. 13-20, (1995)
- [Duc03] D.S.C. Ducci. "Compression of 3D Models with NURBS." (2003).
- [Eva96A] F. Evans, S. Skiena, A. Varshney, "Completing Sequential Triangulations is Hard." Technical Report, Department of Computer Science, State University of New York at Stony BROOKE, (1996)
- [Eva96B] F. Evans, S. Skiena, A. Varshney, "Optimizing triangle strips for fast rendering." IEEE Visualization, pp. 319-326, (1996)
- [Fol96] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. "Computer Graphics: Principles and Practice (Second Edition in C)." Addison-Wesley Publishing Company, (1996)
- [Gal09] R. Gal, O. Sorkine, N. J. Mitra, and D. Cohen-Or. "iWIRES: an analyze-and-edit approach to shape manipulation." ACM Transaction on Graphics Vol. 28, No. 3, Article 33, (2009)

- [Gao06] K. Gao, H. Park, A. Rockwood, and D. Sower. "Attribute based interfaces for geometric modeling". In Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames (Sandbox '06). (2006)
- [Ger92] A. Gersho, R. M. Gray, "Vector Quantization and Signal Compression." Kluwer Academic Publishers, Dordrecht, (1992)
- [Gie98] T. S. Gieng, B. Hamann, K. I. Joy, G. L. Schussman, I. J. Trotts, "Constructing hierarchies for triangle meshes." IEEE Transactions on Visualization on Computer Graphics. Vol. 4, No. 2, pp. 145-161, (1998)
- [Gol09] A. Golovinskiy, J. Podolak, and T. Funkhouser. "Symmetry-Aware Mesh Processing." In Proceedings of the 13th IMA International Conference on Mathematics of Surfaces XIII, pp.170-188, (2009)
- [Gué98] A. Guéziec, G. Taubin, F. Lazarus, W. Horn, "Converting sets of polygons to manifold surfaces by cutting and stitching." IEEE Visualization, pp. 383-390 (1998)
- [Gum98] S. Gumhold, W. StraBer, "Real time compression of triangle mesh connectivity." ACM SIGGRAPH, pp. 133-140, (1998)
- [Gum99] S. Gumhold, "Improved cut-border machine for triangle mesh compression." Erlangen Workshop '99 on Vision, Modeling and Visualization, IEEE Signal Processing Society, (1999)
- [Han98] J. Han and A. Requicha. "Feature recognition from CAD models." IEEE Computer Graphics & Applications, pp. 80-94, (1998)
- [Hec97] P. Heckbert, M. Garland, "Survey of polygonal surface simplification algorithms." Multiresolution Surface Modeling Course Notes of SIGGRAPH '97, (1997)
- [Ise00A] M. Isenburg, "Triangle strip compression." Proceedings of the Graphics Interface, pp. 197-204, (2000)

- [Ise00B] M. Isenburg and J. Snoeyink. "Face Fixer: Compressing Polygon Meshes With Properties." ACM SIGGRAPH Conference Proceedings, pp. 263-270, (2000)

- [Ise02] M. Isenburg. "Compressing Polygon Mesh Connectivity with Degree Duality Prediction." Graphics Interface Conference Proceedings, pp. 161-170, (2002)

- [Jia94] X. Y. Jiang, H. Bunke. "Symmetries of Polyhedra: Detection and Applications." Institut für Informatik und angewandte Mathematik. (1994)

- [Kaz04] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. "Symmetry descriptors and 3D shape matching." Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing (*SGP '04*) (2004).

- [Kho02] A. Khodakovsky, P. Alliez, M. Desbrun, and P. Schröder. "Near-Optimal Connectivity Encoding of 2-Manifold Polygon Meshes." Graphical Models, special issue, (2002)

- [Kin99A] D. King, J. Rossignac, and A. Szmek. "Compression for Irregular Quadrilateral Meshes." Technical Report TR. 99 36, GVU, Georgia Tech, (1999)

- [Kra08] V. Kraevoy, A. Sheffer, A. Shamir, and D. Cohen-Or. "Non-homogeneous resizing of complex models." ACM Transactions on Visualization and Computer Graphics. Vol. 27, No.5, (2008)

- [Kro01] B. Kronrod and C. Gotsman. "Efficient Coding of Non-Triangular Mesh Connectivity." Graphical Models. pp. 263-275 (2001).

- [Lag05] A. Lagae, O. Dumont, and P. Dutre. "Geometry Synthesis by Example. " Proceedings of the International Conference on Shape Modeling and Applications 2005 (SMI '05). IEEE Computer Society, pp. 176-185, (2005)

- [Lee00] E-S. Lee, H.-S. Ko, "Vertex data compression for triangular meshes."

Pacific Graphics, (2000)

- [Lee02] II. Lee, P. Alliez, and M. Desbrun. "Angle-Analyzer: A Triangle-Quad Mesh Codec." Eurographics Conference Proceedings, pp.383 - 392, (2002)
- [Li06] M. Li, F. C. Langbein, and R. R. Martin. "Constructing regularity feature trees for solid models." In Proceedings of the 4th international conference on Geometric Modeling and Processing (GMP'06), Springer-Verlag, pp. 267-286, (2006)
- [Li98] J.Li and C.-C, Jay Kuo. "Mesh Connectivity by the Dual Graph Approach.". MPG98 Contribution Document No. M3530 (1998)
- [Lip08] M. Lipp, P. Wonka, and M. Wimmer. "Interactive visual editing of grammars for procedural architecture." ACM SIGGRAPH 2008 papers. (2008)
- [Lit01] N. Litke, A. Levin, and P. Schöder. "Fitting subdivision surfaces." Proceedings of the conference on Visualization 2001. IEEE Computer Society, pp. 319-324, (2001)
- [Lue01] D .P. Luebke, "A developer's survey of polygonal simplification algorithms." IEEE Computer Graphics Applications Vol. 21, No.3 pp. 24-35 (2001)
- [Man98] M. Mantyla. "An Introduction to Solid Modeling." Computer Science Press (1998)
- [Mar06] A. Martinet, Cyril Soler, N. Holzschuch, and F. X. Sillion. "Accurate detection of symmetries in 3D shapes." ACM Transactions on Graphics Vol. 25, No. 2, pp. 439-464, (2006)
- [MAYA] Autodesk Maya. <http://www.usa.autodesk.com/Maya/>

- [Meh09] R. Mehra, Q. Zhou, J. Long, A. Sheffer, A. Gooch, N. Mitra, "Abstraction of man-made shapes." *ACM Transaction on Graphics* Vol. 28, No. 5, Article 137, (2009)
- [Mer07] P. Merrell. "Example-based model synthesis." *Proceedings of the 2007 symposium on Interactive 3D graphics and games (I3D '07)*. ACM, New York, NY, USA, pp. 105-112, (2007)
- [Min93] P. Minovic, S. Ishikawa, and K. Kato. "Symmetry Identification of a 3-D Object Represented by Octree." *IEEE Transactions on Pattern Anaysis and Machine Intelligence* Vol. 15, No.5, pp. 507-51, (1993)
- [Mit06] N. J. Mitra, L. J. Guibas, and M. Pauly." Partial and approximate symmetry detection for 3D geometry." *ACM Transactions on Graphics* Vol. 25, No. 3, pp. 560-568, (2006)
- [MPEG4-V01] Coding of Audio-Visual Objects: Visual. ISO/IEC 14496-2
- [Mül06] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and Luc Van Gool. "Procedural modeling of buildings." In *ACM SIGGRAPH 2006 Papers (SIGGRAPH '06)*. ACM, New York, NY, USA, pp. 614-623, (2006)
- [Mül07] P. Müller, G. Zeng, P. Wonka, and L. Gool." Image-based procedural modeling of facades." *ACM Transactions on Graphics* Vol. 26, No. 3, Article 85, (2007)
- [Nea07] A. Nealen, Tk. Igarashi, O. Sorkine, and M. Alexa. "FiberMesh: designing freeform surfaces with 3D curves." *ACM SIGGRAPH 2007 papers (SIGGRAPH '07)*. ACM, New York, NY, USA, Article 41, (2007)
- [Nut11] S.Nutanong, E. H.Jacox H.Samet. "An Incremental Hausdorff Distance Calculation Algorithm." In *Proceedings of the VLDB Endowment*, Vol. 4, No. 8, (2011)

- [Par01] Y. I. H. Parish and P. Müller. "Procedural modeling of cities." Proceedings of the 28th annual conference on Computer graphics and interactive techniques (SIGGRAPH '01). ACM, New York, NY, USA, pp. 301-308, (2001)
- [Pau08] M. Pauly, N. J. Mitra, J. Wallner, H. Pottmann, and L J. Guibas. "Discovering structural regularity in 3D geometry." ACM Transactions on Graphics Vol. 27, No.3, (2008)
- [Pen05] J. Peng, C. Kim, and C. C. J Kuo. 2005. "Technologies for 3D mesh compression: A survey." Journal of Visual Communication and Image Representation Vol. 16, No. 6, pp. 688-733, (2005)
- [Pod06] J. Podolak, P. Shilane, A. Golovinskiy, S. Rusinkiewicz, and T. Funkhouser. "A planar-reflective symmetry transform for 3D shapes." ACM Transaction on Graphics Vol. 25, No. 3, pp. 549-559, (2006)
- [Pop97] J.Popovic, H.Hoppe, "Progressive simplicial complexes." ACM SIGGRAPH, pp. 217-224, (1997)
- [Pru96] P. Prusinkiewicz and A. Lindenmayer. The Algorithmic Beauty of Plants. Springer-Verlag New York, Inc., New York, NY, USA. (1996)
- [Raj12] D.S.Rajput, R.R.Kishore. "Error Analysis of 3D Polygonal Model: A Survey." International Journal of Computer Science & Engineering Survey (IJCSSES) Vol. 3, No. 3, (2012)
- [Ros99] J.Rossignac, "Edgebreaker: connectivity compression for triangle meshes." IEEE Transactions on Visualization and Computer Graphics. Vol. 5, No. 1, pp. 47-61, (1999)
- [Rus08] R. M. Rustamov. "Augmented planar reflective symmetry transform." The Visual Computer: International Journal of Computer Grpahics. Vol. 24, No. 6, pp. 423-433, (2008)
- [Sal05] D.Salomon. "Curves and Surfaces for Computer Graphics." Springer-

Verlag NeyWork, Inc., Secaucus, NJ, USA. (2005)

- [Sal06] D.Salomon. "Curves and Surfaces for Computer Graphics." Springer. (2006)
- [Sal09] D. Salomon, G.Motta, D.Bryant. "Handbook of Data Compression." (2009)
- [ShaC04] A. Sharf, M. Alexa, and D. Cohen-Or. "Context-based surface completion." ACM Transaction on Graphics. Vol. 23, No. 3, pp. 878-887, (2004)
- [Shi01] D. Shikhare, S. Bhakar, and S. P. Mudur. Compression of Large 3D Engineering Models using Automatic Discovery of Repeating Geometric Features." Proceedings of the Vision Modeling and Visualization Conference 2001 (VMV '01), pp. 233-240, (2001)
- [Sim06] P. Simari, E. Kalogerakis, and K. Singh. "Folding meshes: hierarchical mesh segmentation based on planar symmetry." In Proceedings of the fourth Eurographics symposium on Geometry processing (SGP '06). Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, pp. 111-119, (2006)
- [Sin98] K. Singh and E. Fiume. "Wires: a geometric deformation technique." In Proceedings of the 25th annual conference on Computer graphics and interactive techniques (SIGGRAPH '98). ACM, New York, NY, USA, pp. 405-414, (1998)
- [Spe97] B.Speckmann, J.Snoeyink, "Easy triangle strips for tin terrain models." In Proceedings of 9th Canadian Conference on Computational Geometry. pp. 239-244 (1997).
- [Sta10] O.Stava, B.Benes, R.Mech, D.G.Aliaga, P.Kristof." Inverse Procedural Modeling by Automatic Generation of L-Systems." Computer Graphics Forum pp. 665-674, (2010)
- [Szy00] A. Szymezak, D. King, J. Rossignac, "An edgebreaker-based efficient compression scheme for regular meshes." Proceedings of 12th Canadian

Conference on Computational Geometry, pp. 257-264, (2000)

- [Tan08] J. W. Tangelder and R. C. Veltkamp. "A survey of content based 3D shape retrieval methods." *Multimedia Tools and Applications*. Vol. 39, No. 3, pp. 441-471, (2008)
- [Tau95] G. Taubin, "A signal processing approach to fair surface design", *ACM SIGGRAPH*, pp. 351-358, (1995)
- [Tau98A] G. Taubin, J. Rossignac, "Geometric compression through topological surgery", *ACM Transaction on Graphics*. Vol. 17, No. 2, pages 84-115, (1998)
- [Tau98B] G. Taubin, W. Horn, F. Lazarus, J. Rossignac, "Geometry coding and VRML." *Proceedings of the IEEE* Vol. 96, No. 6, pp. 1228-1243, (1998)
- [Tau98C] G. Taubin, A. Gueziec, W. Horn, and F. Lazarus. "Progressive forest split compression." *SIGGRAPH 98*, pp. 123-132, (1998)
- [Tou98] C. Touma, C. Gotsman, "Triangle mesh compression." *Proceedings of Graphics Interface*, pp. 26-34, (1998)
- [Tur84] G. Turan, "On the succinct representations of graphs." *Discrete Applied Mathematics* Vol. 8 pp. 289-294, (1984)
- [Van93] J. Vandenbrande and A. Requicha. "Spatial reasoning for the automatic recognition of machinable features in solid models." *IEEE Pattern Analysis and Machine Intelligence*, Vol. 15, No. 12, pp. 1269-1285, (1993)
- [VRML97] The Virtual Reality Modeling Language (VRML). *ISO/IEC 13772-1*, 1997.
- [Wan11] Y. Wang, K. Xu, J. Li, H. Zhang, A. Shamir, L. Liu, Z. Cheng, Y. Xiong, "Symmetry Hierarchy of Man-Made Objects," *Computer Graphics Forum (Special Issue of Eurographics)* (2011)

- [Won03] P. Wonka, M. Wimmer, F. Sillion, and W. Ribarsky. "Instant architecture." *ACM Transaction on Graphics*. Vol. 22, No. 3, pp. 669-677, (2003)
- [Xia99] X. Xiang, M. Held, J. Mitchell, "Fast and efficient stripefication of polygonal surface models." *ACM Symposium on Interactive 3D Graphics*, pp. 71-78, (1999)
- [Zor00] D. Zorin, P. Schroder. "Subdivision for Modeling and Animation." *Siggraph 2000 Course Notes*.

Appendix A - Example MPD document

#Vertices

```
V      1.182020068 2.004468441 2.853650093
V      1.544613361 1.996932745 2.675350666
V      1.880322576 2.004468441 2.450485229
V      1.285832644 2.364962101 3.104275703
V      1.680271029 2.357344151 2.910316467
V      2.045464039 2.364962101 2.665702343
V      2.184413433 1.996932745 2.184414864
V      2.450483799 2.004468441 1.880324125
V      2.376262426 2.357344151 2.376263618
V      2.665700912 2.364962101 2.045465708
V      2.675349236 1.996932745 1.544615269
V      2.853649139 2.004468441 1.182021737
V      2.910315275 2.357344151 1.680273056
V      3.10427475 2.364962101 1.285834312
V      2.983964682 1.996932745 0.7995526195
V      3.062309265 2.004468441 0.4034208655
V      3.246035099 2.357344151 0.8697744012
V      3.331260443 2.364962101 0.4388517439
V      3.089227438 1.996932745 2.044337407e-006
V      3.062342644 2.004468441 -0.4031630158
V      3.360542774 2.357344151 2.161711336e-006
V      3.331296921 2.364962101 -0.4385714531
```

V 2.98396492 1.996932745 -0.7995486259
V 2.853649378 2.004468441 -1.182018042
V 3.246035576 2.357344151 -0.8697701097
V 3.104275227 2.364962101 -1.285830498
V 2.675349951 1.996932745 -1.544611216
V 2.450484753 2.004468441 -1.880320311
V 2.91031599 2.357344151 -1.680268764
V 2.665701866 2.364962101 -2.045461893
V 2.184414864 1.996932745 -2.184411049
V 1.88032341 2.004468441 -2.450481892
V 2.376263618 2.357344151 -2.376260281
V 2.045465469 2.364962101 -2.665698528
V 1.544614553 1.996932745 -2.67534709
V 1.182021141 2.004468441 -2.853646994
V 1.680272341 2.357344151 -2.910312891
V 1.285833836 2.364962101 -3.104272366
V 0.7995521426 1.996932745 -2.983962297
V 0.4031665921 2.004468441 -3.062340736
V 0.8697738647 2.357344151 -3.246032715
V 0.4385751486 2.364962101 -3.331294775
V 1.462495902e-006 1.996932745 -3.089225292
V -0.4034177065 2.004468441 -3.062307596
V 1.552026788e-006 2.357344151 -3.360540628
V -0.4388484359 2.364962101 -3.331258535
V -0.7995491028 1.996932745 -2.983963013
V -1.182018518 2.004468441 -2.853647709
V -0.8697707653 2.357344151 -3.246033192
V -1.285830975 2.364962101 -3.104272842

V -1.54461205 1.996932745 -2.675347805
 V -1.880321145 2.004468441 -2.450482368
 V -1.68026948 2.357344151 -2.910313606
 V -2.045462847 2.364962101 -2.665699482
 V -2.184411287 1.996932745 -2.184412479
 V -2.450481892 2.004468441 -1.880321503
 V -2.376260281 2.357344151 -2.376261234
 V -2.665699005 2.364962101 -2.045463085
 V -2.675347805 1.996932745 -1.544612646
 V -2.853647232 2.004468441 -1.182019234
 V -2.910313606 2.357344151 -1.680270433
 V -3.104272604 2.364962101 -1.285831809
 V -2.983962774 1.996932745 -0.7995500565
 V -3.062340736 2.004468441 -0.4031643867
 V -3.246033192 2.357344151 -0.869771719
 V -3.331295013 2.364962101 -0.4385728538
 V -3.089225292 1.996932745 4.663657762e-007
 V -3.062340736 2.004468441 0.403165549
 V -3.360540628 2.357344151 4.495504982e-007
 V -3.331295013 2.364962101 0.4385740757
 V -2.98396349 1.996932745 0.7995511293
 V -2.853647947 2.004468441 1.182020545
 V -3.246034145 2.357344151 0.8697727323
 V -3.104273558 2.364962101 1.28583312
 V -2.675348282 1.996932745 1.544613838
 V -2.450482845 2.004468441 1.880323052
 V -2.910314083 2.357344151 1.680271626
 V -2.665700197 2.364962101 2.045464993

V -2.184412479 1.996932745 2.18441391
 V -1.880321622 2.004468441 2.450484276
 V -2.376261473 2.357344151 2.376262903
 V -2.045463324 2.364962101 2.665701151
 V -3.74777317 1.710401058 9.047922134
 V -3.171112061 2.004468441 7.655739307
 V -2.050371885 2.004468441 4.950034618
 V -1.544613242 1.996932745 2.675349712
 V -1.182020068 2.004468441 2.853649378
 V -4.470392227 -0.9550725222 10.79247665
 V -4.304643154 -0.1324029267 10.39232254
 V -4.083518982 1.32356751 9.858481407
 V -4.541399002 -0.8333937526 10.96390152
 V -4.547614098 -0.9100540876 10.97890663
 V -4.388273716 -0.04221072048 10.59422493
 V -4.163923264 1.428946972 10.05259609
 V -4.069412231 2.065053463 9.824426651
 V -3.449619293 2.364962101 8.328115463
 V -2.230448246 2.364962101 5.384778023
 V -1.68027091 2.357344151 2.910315514
 V -1.285832644 2.364962101 3.104274988
 V -4.462006569 -0.9134507775 10.77223206
 V -3.701694727 1.175346017 10.29519272
 V -3.250249863 -0.438547641 10.24209023
 V -2.72795248 0.1112863272 10.18085289
 V -3.19336009 0.1219652295 10.04195023
 V -3.369077682 -0.8176879883 10.60052681
 V -2.804794312 1.417075515 10.46762562

V -3.374131918 -0.854424417 10.61523628
 V -1.941838622 1.175346017 10.76674461
 V -2.306250095 -0.438547641 10.49503422
 V -2.255448103 0.1219652295 10.29326153
 V -2.38256073 -0.8176879883 10.86486244
 V -2.385538578 -0.854424417 10.88012886
 V -2.738225698 1.313113928 10.21918583
 V -3.672486782 1.119760513 10.06726837
 V -3.289315224 -0.5482262373 10.03633118
 V -2.666763544 0.1105846986 9.952493668
 V -3.217802048 0.1120592207 9.807010651
 V -3.402665377 -0.9261751175 10.40357208
 V -1.853171229 1.119760752 10.55475235
 V -2.116810322 0.1120592728 10.10201836
 V -2.169539213 -0.5482260585 10.33637428
 V -2.254995823 -0.9261767864 10.71108913
 V -1.27829659 1.710401058 9.709616661
 V -1.081608295 2.004468441 8.215620041
 V -0.6993441582 2.004468441 5.312041759
 V -0.7995506525 1.996932745 2.98396492
 V -0.403165102 2.004468441 3.062343121
 V -1.524769306 -0.9550725222 11.58175564
 V -1.468235135 -0.1324029267 11.15233707
 V -1.392813683 1.32356751 10.57945442
 V -1.548988104 -0.8333937526 11.76571655
 V -1.551108003 -0.9100540876 11.78181839
 V -1.49676013 -0.04221072048 11.3690033
 V -1.420238256 1.428946972 10.7877655

V -1.388002157 2.065053463 10.54290962
 V -1.176601887 2.364962101 8.937168121
 V -0.7607648373 2.364962101 5.778578758
 V -0.8697722554 2.357344151 3.246035576
 V -0.4385735989 2.364962101 3.331297398
 V -3.396930695 -0.9070533514 10.38499165
 V -2.250672102 -0.9070549607 10.69213104
 V -1.521908998 -0.9134507775 11.56002998
 V 6.337111813e-008 1.996932745 3.089227676
 V 0.4031651616 2.004468441 3.062343597
 V 3.441985541e-008 2.357344151 3.360543251
 V 0.4385736585 2.364962101 3.331297874
 V 0.7995505929 1.996932745 2.983965158
 V 0.8697721362 2.357344151 3.246035814

#Curves

C 98 137 138
 C 138 136 135 134 133
 C 133 105 93
 C 93 94 95 96 98
 C 133 132 130 131 127 141 128 129
 C 129 112 89
 C 89 88 99 87 91 90 92
 C 103 102 110
 C 133 132 130 131 127 141 128 129
 C 119 115 116
 C 116 115 139 117 106 104 101 103
 C 133 110

C 119 129
C 116 89
C 103 93
C 89 82 83 85 86
C 127 124 123 131 129
C 86 125 127

C 139 145 146
C 146 148 4
C 4 5 6
C 6 9 10
C 10 13 14
C 14 17 18
C 18 21 22
C 22 25 26
C 26 29 30
C 30 33 34
C 34 39 38
C 38 41 42
C 42 45 46
C 46 49 50
C 50 53 54
C 54 57 58
C 58 51 62
C 62 65 66
C 66 69 70
C 73 74 77
C 77 78 81

C 81 82 98

C 128 127 85

C 85 86 80

C 80 79 76

C 76 75 72

C 72 71 68

C 68 67 64

C 64 63 60

C 60 59 56

C 56 55 52

C 52 51 48

C 48 47 44

C 44 43 40

C 40 39 36

C 36 35 32

C 32 31 28

C 28 27 24

C 24 23 20

C 20 19 16

C 16 15 12

C 12 11 8

C 8 7 3

C 3 2 1

C 1 147 144

C 144 143 128

```

#Patches

#ROT {copies} {deg} {axis: x y z}, {origin: x y z}

ROT 11 30 0 1 0 0 0 0{

    P 1 2 3 4

    P -3 -12 -8 15

    P 5 -13 -9 12

    P 8 9 10 11

    P 13 6 -14 -10

    P 14 7 -15 -11

    P 16 18 17 -6

}

P R 1 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40

P R 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62

63 64

```