การลดทอนรายละเอียดเมชแบบยุบโดยใช้การตั้งค่าคะแนนด้วยการเบี่ยงเบนเชิงมุม และความปรกติของหน้า

นาย วรากร อึ้งวิเชียร

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2553

COLLAPSE-BASED MESH SIMPLIFICATION USING ANGULAR DEVIATION AND REGULARITY BIAS

Mr. Varakorn Ungvichian

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy Program in Computer Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2010

Thesis Title	COLLAPSE-BASED MESH SIMPLIFICATION USING ANGULAR DEVIATION AND REGULARITY BIAS
By	Mr. Varakorn Ungvichian
Field of Study	Computer Engineering
Thesis Advisor	Assistant Professor Pizzanu Kanongchaiyos, Ph.D.

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirements for the Doctoral Degree

[illegible] .... Dean of the Faculty of Engineering
(Associate Professor Boonsom Lerthiranwong, Dr.Ing.)

THESIS COMMITTEE

[illegible] .... Chairman
(Associate Professor Somchai Prasitjutrakul, Ph.D.)

[illegible] .... Thesis Advisor
(Assistant Professor Pizzanu Kanongchaiyos, Ph.D.)

Chakrit Watcharopas .... External Examiner
(Chakrit Watcharopas, Ph.D.)

[illegible] .... External Examiner
(Assistant Professor Natasha Dejdumrong, Ph.D.)

[illegible] .... External Examiner
(Associate Professor Pavadee Sompagdee)

วรากร อึ้งวิเชียร : การลดทอนรายละเอียดเมชแบบยุบโดยใช้การตั้งค่าคะแนนด้วยการเบี่ยงเบนเชิงมุมและความปรกติของหน้า. (COLLAPSE-BASED MESH SIMPLIFICATION USING ANGULAR DEVIATION AND REGULARITY BIAS) อ. ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ. ดร. พิษณุ คนองชัยยศ, 199 หน้า.

หัวข้อวิจัยคอมพิวเตอร์กราฟิกส์ที่สำคัญคือ การลดทอนเมช หรือการลดจำนวนหน้าของโมเดลสามมิติที่ซับซ้อน เพื่อเพิ่มสมรรถภาพในการเรนเดอร์โดยที่ยังคงคุณภาพของรูปภาพ งานวิจัยในปัจจุบันจะใช้วิธีการที่ใช้การยุบเส้นขอบ เช่น ค่า error metric ยกกำลังสองของ Garland และ Heckbert เนื่องจากวิธีการเหล่านี้สามารถใช้ได้ดีกับโครงสร้างข้อมูลที่เก็บระดับความละเอียด ได้มีผลงานวิจัยที่แสดงการปรับปรุงวิธีการของ Garland และ Heckbert โดยใช้คะแนนที่มีฐานจากความโค้ง อย่างไรก็ดี การใช้ความโค้งสำคัญ (principal curvature) ทั้งสองค่า รวมทั้งทิศทาง สามารถลดความคลุมเครือที่เกิดจากการใช้คะแนนค่าเดียวได้ การปรับปรุงวิธีการเดิมของ Garland และ Heckbert ที่นำเสนอคำนวณหาค่าของความโค้งสำคัญและทิศทางของแต่ละเวอร์เท็กซ์ เพื่อคำนวณหาค่าสัมบูรณ์ของค่าความโค้งเส้นปกติในทิศทางขอบที่ลด นอกจากนี้ มีการใช้ความปรกติของและการเบี่ยงเบนของมุมของหน้าที่ได้ เพื่อคำนวณคะแนนโทษด้วย และมีการอธิบายถึงวิธีการปรับค่าในฮีพที่ปรับค่าเฉพาะส่วนบนสุด เพื่อลดเวลาที่ใช้ในการทำงาน ได้สังเกตว่า วิธีการใหม่ทำให้ได้ค่าเฉลี่ยของระยะ Hausdorff ซึ่งใช้ในการวัดค่าต่างของเมช ที่น้อยกว่า QEM ในช่วง 12% ถึง 70% ระหว่างช่วง 5% ถึง 50% ของจำนวนหน้าเดิม อย่างไรก็ดี QEM ยังให้ระยะที่น้อยกว่า เมื่อลดเป็นจำนวนหน้าที่น้อยกว่า อัลกอริทึมที่เสนอยังคงมีระยะเวลาที่ใช้ $O(n \log n)$ แต่ วิธีการปรับปรุงฮีพบางส่วนสามารถเพิ่มความเร็วของกระบวนการถึง 5.4 เท่าเมื่อเทียบการการปรับปรุงฮีพทั้งหมด

ภาควิชา......วิศวกรรมคอมพิวเตอร์...... ลายมือชื่อนิสิต [illegible]
สาขาวิชา......วิศวกรรมคอมพิวเตอร์...... ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก [illegible]
ปีการศึกษา 2553

5071821021 : MAJOR COMPUTER ENGINEERING
KEYWORDS : THREE DIMENSIONAL COMPUTER GRAPHICS / MESH SIMPLIFICATION / CURVATURE / EDGE CONTRACTION

VARAKORN UNGVICHIAN : COLLAPSE-BASED MESH SIMPLIFICATION USING ANGULAR DEVIATION AND REGULARITY BIAS. ADVISOR : ASST. PROF. PIZZANU KANONGCHAIYOS, PH.D., 199 pp.

A major research topic in computer graphics is mesh simplification, reducing the face count of complex 3D models to improve rendering performance while retaining visual quality. Current research prefers edge contraction based methods, such as Garland and Heckbert's Quadric Error Metric, as such methods lend themselves well to level-of-detail structures. Various research has suggested improvements to QEM based on curvature-based scoring; however, using the two principal curvatures and their directions can help reduce the inherent ambiguity of using a single score. The proposed extension to Garland and Heckbert's method calculates the principal curvatures and their directions for each vertex, to calculate the absolute normal curvature in the direction of contraction. Also, the regularity and the angular and dihedral deviations of the resulting faces are used to apply penalties. A heap updating scheme that only updates the top portion of the heap to save time is also described. The proposed method has been observed to reduce the average Hausdorff distance, a measure of mesh difference, in a range between 12%-70% from 5% to 50% face count, although QEM still produces lower distances at lower face count. Although the proposed algorithm retains an $O(n \log n)$ time complexity, the partial heap update scheme has improved the overall process by a factor of 5.4 compared to using full heap updates.

Department : Computer Engineering	Student's Signature [illegible]
Field of Study : Computer Engineering	Advisor's Signature [illegible]
Academic Year : 2010	

ACKNOWLEDGEMENTS

First of all, I thank my thesis advisor, Assistant Professor Pizzanu Kanongchaiyos, Ph.D., for his useful advice and research suggestions, as well as the thesis committee for this project for their useful feedback: Associate Professor Somchai Prasitjutrakul, Ph.D., Assistant Professor Pizzanu Kanongchaiyos, Ph.D., Chakrit Watcharopas, Ph.D., Assistant Professor Natasha Dejdumrong, Ph.D., and Associate Professor Pavadee Sompagdee.

I also thank the International Association of Computer Science & Information Technology, for accepting my research paper for presentation at the 2nd International Conference on Computer Modeling and Simulation, in Sanya, China, between January 22-24, 2010.

Lastly, I also have to thank my family for their love and support through all my years of education.

CONTENTS

Page

Page

LIST OF TABLES

Table | Page

LIST OF FIGURES

Page

Page

Page

CHAPTER I

INTRODUCTION

In this chapter, we will explain the problem statement and motivation behind our research into mesh simplification. We will then describe the contributions we have made in this dissertation, and then provide an outline of the structure of the remainder of the dissertation.

1.1 Problem Statement and Motivation

One of the many fields of computer research is computer graphics, which can be used to generate imagery in two to three dimensions, allowing for many applications, such as for simulation, gaming, or animation. Because three-dimensional graphics are more suitable for representing real-world situations, especially for simulations, three-dimensional graphics have become more popular. As the number of applications for computer graphics has increased, along with computer graphics hardware capability in general, so has the quality of three-dimensional polygonal models.

Current models generally have on the order of hundreds of thousands, and sometimes even millions of faces. For example, Stanford University has produced a model of Michelangelo's David statue containing about one billion faces, with models of other statues by Michelangelo also available at a detail on the level of hundreds of millions of faces (Levoy et al., 2000). Such a high number of faces makes current 3D models not only more detailed and realistic, but also more complex than can be efficiently rendered on typical graphics hardware, which has an inherent limit on the speed of rendering.

In many applications, such fine detail is unnecessary and impractical, especially when real-time display is desired, for example, in online role playing games, where a high number of characters may appear on screen at a given time, and real-time response to user input is of higher importance than high-quality rendering. Therefore, it is usually necessary to reduce the number of faces in the model to a much smaller number, while retaining as much visual (and/or geometric) resemblance to the original model as possible. This process is known as *mesh simplification* or *mesh reduction*, and many methods have been proposed in the literature, using several different schemes. However, one of the most well-known and widely used algorithms is Garland and Heckbert's Quadric Error Metric method (Garland and Heckbert, 1997), or QEM, which scores contractions based mainly on the sums of the squares of the distances of the resulting vertex from the original mesh's faces.

As Garland and Heckbert's original metric is based solely on geometric error, many subsequent papers have proposed enhancements to the original algorithm to improve its performance. For example, some papers (Kim et al., 2008) have described methods to efficiently extend the original metric to meshes with vertex color, while others have incorporated other factors, such as curvature-based factors (Xu et al., 2008; Li and Zhu, 2008), torsion (Jong et al., 2006), normal variance (Choi, 2008), and higher-dimension feature sensitivity metrics (Wei and Lou, 2010) to allow the algorithm to recognize visually important features, and thus improve the resulting simplified model. Assuming that there is an upper bound to the number of faces adjacent to each vertex, edge-contraction based simplification algorithms can generally be shown to have an overall time complexity of $O(n \log n)$ (Garland and Heckbert, 1997). The current state of the art, however, has suggested a possible ambiguity in the measure of curvature and curvedness.

The formal definition of curvature on a surface involves two directions and their respective normal curvatures, which are the maximum ($k_{\max}$) and minimum ($k_{\min}$) curvatures of the local area around a given point (Batagelo and Wu, 2007). The Gauss curvature K is the product of the maximum and minimum, while curvedness is measured with a Pythagorean sum of the two curvatures ($\sqrt{k_{\min}^2 + k_{\max}^2} / \sqrt{2}$). Although both of these measures are easy to calculate, each one results in the possibility of surfaces of different nature receiving the same measure, which is to be expected of measures that provide a single value to determine the curvedness of the surface. Our expectation is that the quality of the simplified mesh can be improved by taking both principal curvatures and directions into consideration. The principal curvatures indicate the minimum and maximum normal curvature around a given point on the surface, and we believe that these values can help indicate the direction in which it is more suitable to move a vertex, as we expect that moving vertices in directions of little to no curvature (either on a flat surface or along a straight feature edge) would affect the overall structure less than moving them in high curvature directions.

The angular orientation of faces affects their rendered appearance, for example, the reflection of light. Therefore, many contraction-based techniques check the normal vectors of the resulting faces to prevent faces from folding over by penalizing such contractions (such as QEM), while other methods use the angular deviation as part of the score calculation. Hussein et al. (2001) and Choi (2008) use the angular deviation from the current facial orientation as part of their respective approaches, while other existing algorithms, such as Kobbelt (1998), focus on the overall dihedral angle between pairs of faces after contraction, instead of the change in the angle. However, Jia et al. (2006) point out a shortcoming with the latter method: A feature edge between faces with already large dihedral angles would produce a high contraction score, even in cases where contracting the edge would not significantly alter the model's appearance (for example, an edge between two facets of a cube). We also believe that considering overall dihedral angular deviation between faces, that is, the relative orientation between a given pair of faces, can be used to better improve overall orientation, by preventing features from flipping between concavity and convexity.

Another less-researched topic about contraction-based algorithms is the process of updating the heap. Most algorithms are assumed to perform updates on the full heap after each contraction, whereas some papers have performed research into improving the efficiency of heap updates. Cohen et al. (1997) flag scores for later updates, while Wu and Kobbelt (2004) do away with the heap by using a random-choice method that picks edges at random and contracts the best-scoring one, and Chen et al. (2004) propose a method that saves memory space for the heap by setting a limit on its size, filling the heap to the size limit, and then alternating edge contractions with edge scoring, with a claim of linear time complexity due to the constant heap size limit. The methods have been shown to not significantly affect the accuracy of the simplification. However, we feel that there should be some degree of assurance that a good-scoring edge is being contracted at each step, and that progress is being made in the simplification process.

1.2 Contributions

The algorithm that we propose in this research uses a vertex-merging approach based on QEM, utilizing the curvature direction to avoid ambiguity of curvature measures, and angular and dihedral deviation to control facial orientation. It has boundary preservation, and uses Kovalevsky's topological data representation as its input in order to reduce the time to pre-calculate necessary topological relationships. The algorithm is of a dynamic approach to mesh reduction, allowing for an exact level-of-detail to be defined offline, without considering the rendering viewpoint at runtime. It also uses a "lazy" method of heap updating to reduce runtime, at little to no cost of accuracy.

Our algorithm produces lower Hausdorff distances than QEM during the early stages of the simplification on average (up to 5% level of simplification). However, at drastic levels, it produces significantly worse distances on average.

1.3 Dissertation structure

The remainder of the paper will be sectioned as follows: Chapter II (Related Work) will describe the details of prior research that relates to the work presented here. Chapter III (The Proposed Algorithm) will describe the workings of the algorithm. Chapter IV (Experiment and Results) will describe the experiments and results, and discuss our results. Lastly, Chapter V (Conclusion) will suggest possible further improvements.

CHAPTER II

BACKGROUND AND RELATED WORK

This chapter provides background information and details on previous work related to this current research. We will explain the basics of polygonal meshes and data structures used to represent them, before discussing previously published research work about edge-contraction based mesh simplification methods and heap updating methods. We conclude the chapter by explaining the aims of our research in context of the previous work.

2.1 Basics of polygonal meshes

There are many methods to represent three-dimensional models, such as boundary representation and solid modeling; however, the most common method is to use a *polygonal mesh*, a structure comprised of a collection of vertices $\{v_0, v_1, v_2...\}$, edges $\{e_0, e_1, e_2...\}$ and faces $\{f_0, f_1, f_2...\}$, representing the surface of the model. Although the faces can be of any shape, triangles are usually used, as the surface of a triangle can be shown to be planar, and thus consistently-defined; in fact, many rendering systems will decompose non-triangular faces into triangles before display.

Another issue about the storage of three-dimensional models is the data structure used to represent the models. Many data structures have been developed for this purpose, each of which has its own advantages and disadvantages (Smith, 2006). Among the most notable examples of such data structures are:

Vertex-vertex meshes: Object is represented as a list of vertices adjacent to other vertices. Simple, but all edges and faces are implicit.

Face-vertex meshes: Object is represented as a list of faces and vertices. Most generally used (e.g., the Object File Format or OFF), but edges are still implicit.

Winged-edge meshes (Baumgart, 1975): Object is represented explicitly as a list of faces, edges and vertices, along with topological relationships. Flexible for geometric operations, but only allows for *manifold meshes* (meshes where every edge is adjacent to exactly two faces), due to specifying the nearest clockwise and counterclockwise edge at each endpoint for each edge (which only makes sense on fully manifold meshes).

Most mesh simplification algorithms do not specify a representation for the input data, and generally assume that the data input into the algorithm will contain the necessary relationships, or that they can be derived at trivial cost. A few examples of papers that take the representation of the input data into consideration are:

Lindstrom's Out Of Core method (Lindstrom, 2000): In addition to general face-vertex meshes, this paper claims the ability to work on a "triangle soup", in which each individual triangle is represented directly as a triplet of vertex coordinates (at the cost of extra disk space and much redundancy), as its use of a clustering mechanism along with vertex merging means that the algorithm does not require connectivity information.

Vieira’s Fast Stellar Simplification method (Vieira et al., 2003): This paper adapts the “Corner Table” data structure by Rossignac (2001) to represent triangular meshes, essentially a compact version of a half-edge representation, and features an algorithm designed to use this structure. As with the winged-edge structure, the “Corner Table” data structure assumes fully manifold models.

A paper by Kovalevsky (2001) proposes a “cell-list” representation that we have named the Abstract Cellular Complex format, which allows for the explicit storage of all topological relationships (for example, all faces adjacent to a given edge) between mesh elements of various dimensions without requiring a search, thus allowing for all necessary relationships to be pre-calculated and stored. This representation also allows for non-manifold structures to be represented (unlike the winged-edge structure). We have already created an algorithm for reconstructing a possible 3D model from a wireframe into this representation (Varakorn Ungvichian and Pizzanu Kanongchaiyos, 2006), and it is relatively trivial to create when the vertices and faces are already known. Pre-calculating the relationships as part of the structure can assist the mesh simplification process, as necessary relationships no longer have to be determined.

2.2 Mesh simplification

The discrete nature of polygonal meshes means that to store more detail, more faces are required. Due to the inherent limits of graphics hardware, the high face count of a highly-detailed model may result in inefficient rendering. In order to improve rendering performance, many algorithms have been developed to reduce the face count of complex models. The problem of mesh simplification can be formally defined as follows:

For a given polygonal model P, determine a model P' with x faces that has the most resemblance to P.

An example of results from mesh simplification is shown in Figure 2-1, from Cignoni (2003). The model of the David statue (left) has been simplified from 8 million faces to 1.7 million (center) and 10,000 faces (right), while still retaining visual resemblance to the original. The rightmost model is suitable for most casual applications, can be much more quickly rendered, and is also easier to store and transmit over a network.

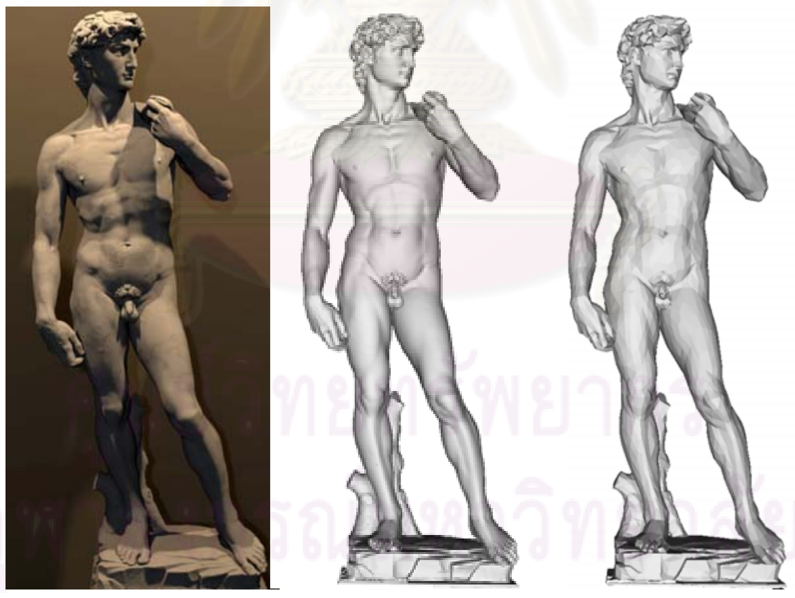


Figure 2-1: David model

Another reason to use mesh simplification is to reduce the detail of objects that are further away from the viewer, as the details will no longer be as visible; therefore, such objects will not require as many faces to represent properly. As an example, Figure 2-2, from Cohen (1996), shows an example of an array of brake assembly rotors. The furthest rotors take up less screen space than those in the foreground, and thus require fewer faces.

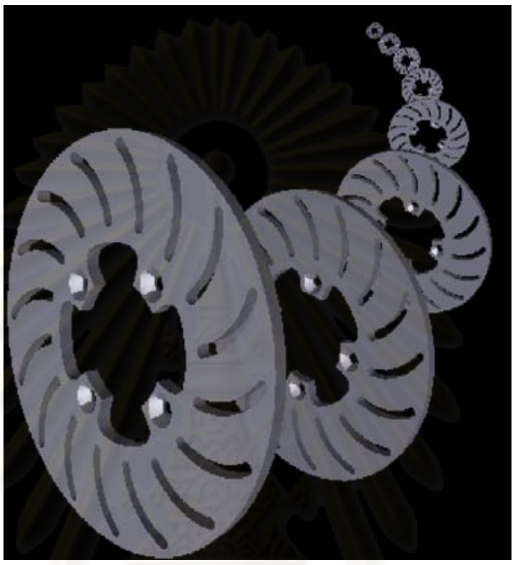


Figure 2-2: Brake assembly rotors in distance

The main research problems concerning mesh simplification are: How to best quantize the resemblance of the simplified model to the original, how to minimize the amount of artifacts and visual errors resulting from the simplification process, and how to maintain a reasonable running time. It should be noted that Agarwal and Suri (1994) have shown that the problem of finding an optimal simplified approximation for a general model is NP-Hard, that is, likely to be solved only through impractical brute force algorithms; and in any case, there may be several ways to define "optimal", such as most visually similar or least surface difference. Therefore, proposed methods use various heuristics to determine an approximation of an optimal simplification. Detailed reviews of various mesh simplification methods can be found in surveys by Cignoni (1997) and Luebke (2001), the latter of which classifies the methods by mechanism, treatment of topology and whether they use static, dynamic, or view-dependent simplification.

According to Luebke (2001), there are four major mechanisms for mesh simplification under which such algorithms can be classified, as illustrated in Figure 2-3: *Sampling,* which involves sampling the geometry either with points on the surface (Boubekeur and Alexa, 2009), or voxels in a grid (Rossignac and Borrel, 1993); a*daptive subdivision*, which involves determining a base mesh from the original model and recursively subdividing it back into the original (Eck et al., 1995); *decimation*, which involves removing vertices and their surrounding faces, before re-triangulating the resulting holes (Schroeder et al., 1992); and *vertex merging*, which involves merging vertices together and removing degenerate faces (Garland and Heckbert, 1997).

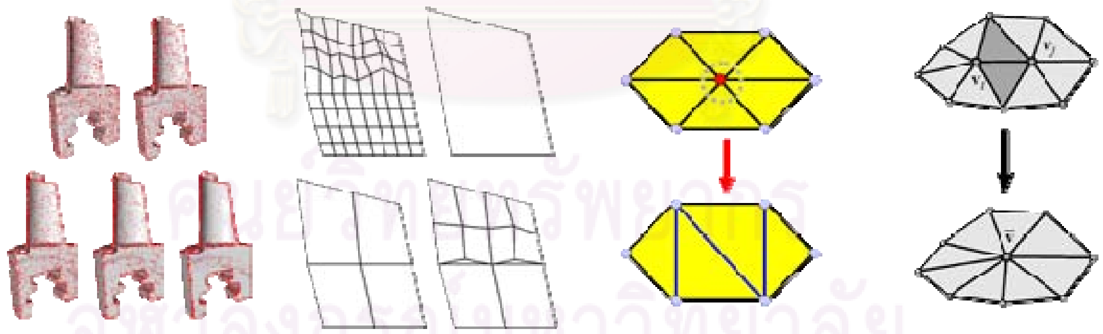


Figure 2-3: Mesh simplification mechanisms (left to right): Sampling, adaptive subdivision, decimation, vertex merging

The most commonly-used and researched mechanism is vertex merging, more specifically, *edge contraction*, in which the two vertices of an edge are reduced to one, thus causing the edge’s adjacent faces to become degenerate, with edge selection based on scoring. The vertex merging mechanism can be extended to contracting triangular faces (Hamann, 1994; Gieng et al., 1997; Zhigeng, Jiaoying and Kun, 2001), or structures with more complexity (Chen et al., 2007), however, it has been noted that such extensions reduce the quality of the simplified model. Examples of methods that use this mechanism include the energy function-based progressive meshes (Hoppe, 1996) and the volume and boundary area-based fast and memory efficient polygonal simplification, aka the Memoryless method (Lindstrom and Turk, 1998). However, one of the most well-known and widely used contraction-based mesh simplification algorithms is Garland and Heckbert's Quadric Error Metric method (Garland and Heckbert, 1997), which scores contractions based mainly on the sums of the squares of the distances of the resulting vertex from the original mesh's faces. Contraction-based methods are well-suited for level-of-detail based structures, such as Hoppe’s progressive meshes, as one can simply store the edge contraction steps as part of the structure, and then use the steps to re-create the mesh at any desired level of detail.

Another one of Luebke’s classifications for mesh simplification algorithms is their treatment and tolerance to topology. *Topology-preserving* algorithms, such as decimation, can preserve manifold connectivity by avoiding contractions that affect the local topology such as holes, and thus can help improve visual fidelity. However, such algorithms also have a limit on the level of simplification that can be performed. In order to produce drastic reductions in face count, the majority of simplification algorithms are *topology-modifying* and *topology insensitive*, as the algorithms are allowed to modify the local topology, usually without taking the initial local connectivity into account. *Topology-tolerant* algorithms can properly handle a mesh that has non-manifold portions. Although three-dimensional models, especially those scanned from real objects, should ideally be manifold at all points, models may have some topological imperfections, such as more than two faces meeting at a single edge. Therefore, topology-tolerant algorithms can be used on any model without first requiring the topology of the model to be verified.

An example of topology preservation is to preserve the boundaries around the edges of a structure, or holes. Such preservation can be promoted, or enforced, using various methods, for example: calculating planes of constraint and using them to calculate an extra quadric-based score (Garland and Heckbert, 1997), penalizing edge contractions that affect boundaries (Fahn et al., 2002), and constraining boundaries to remain within a defined area (Zelinka and Garland, 2002). Although penalizing contractions on boundaries lowers the chances of a contraction significantly affecting them, providing direct constraints on such contractions provides an absolute guarantee of preservation. The cost of performing boundary preservation includes the determination and storage of boundary edges and vertices, and the calculation of penalties for edge contractions involving boundaries.

Luebke’s other categorization method is the type of mesh simplification: *static*, *dynamic*, and *view-dependent*. For most general objects, the current preference for mesh simplification is a dynamic approach, such that a model at any given level of detail can be produced. Of the other types, static mesh simplification allows for only a fixed set of simplified models that may not suit all possible uses of the model, while view-dependent approaches are designed for large objects such as terrains, and requires making simplification decisions at runtime in the renderer, rather than performing the mesh simplification process offline.

2.3 Related work

In our review of related works, we will categorize the works into two categories: QEM-based and non-QEM-based. QEM-based works will be categorized into three subcategories: works based on quadric weighting, score penalizing and expanding the quadric matrices. Non-QEM will be categorized into five subcategories: appearance-preserving approaches, memory-saving approaches, subdivided meshes, global properties and optimal placement. We will also discuss papers that consider the heap updating process

2.3.1 QEM and QEM-based approaches

Although many types of mechanisms exist to reduce the number of faces in a model while retaining as much visual or geometric resemblance to the original as possible, we shall only focus on methods that use edge contraction to achieve the desired simplification. Cignoni (1997) and Luebke (2001) have produced more comprehensive reviews of the state of the art in mesh simplification.

Edge contraction is well-suited for mesh simplification for structures based on level of detail, as one can store the sequence of contractions as part of the structure, from which one can then easily recreate a model at a relatively exact desired level. This property makes level-of-detail-based structures more attractive than previous static structures, as it allows the rendering program to devote more faces to more important objects, such as those in the foreground, while allocating an exact number of faces. The concept of a hierarchical structure for geometric models was originally proposed by Clark (1976), in which he proposes “varying environmental detail” as a possible use.

An early example of an edge contraction-based algorithm and a level of detail-based structure was Hoppe’s Progressive Meshes (Hoppe, 1996), which proposes both a data structure that stores the progression of vertex mappings and edge contractions, and a mesh simplification method to create models using this structure. Their simplification method was based on greedy reduction of an energy function. However, it has been noted that the simplification process is slow and intricate to code. Another more recent example of a data structure for level-of-detail rendering is “masking strips” (Ripolles, 2009), a structure based on triangle strips that uses edge-contraction based mesh simplification as part of the algorithm for creating the structure.

Garland and Heckbert (1997) devised a method for mesh simplification that is still widely used in various forms, known as the Quadric Error Metric method (or QEM). Their method calculates a symmetric 4×4 matrix for each vertex, based on the planes (normal vectors) of its adjacent faces. For each plane *p*:

$$K_p=\begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix} \tag{2-1}$$

where the plane equation of *p* is $ax+by+cz+d=0$ and $|a, b, c|=1$.

The K_p matrix can be used to calculate the square of the distance from a given point v to p: $v^T K_p v$. The plane p (and thus K_p) of a given triangular face can be determined easily using the coordinates of its vertices, and for each vertex, the resulting matrix is a sum (which can be either regular or weighted to reduce the effects of tessellation) of the matrices obtained from the adjacent faces. For each edge, the matrices at each endpoint are summed together and used to determine the vertex position with the least (or lesser) error and calculate the total score, before inserting each score into a heap, and repeatedly contracting edges with the least total score (updating scores as needed) until reaching the desired level of simplification. Figure 2-4 shows examples of the isosurfaces resulting from the quadric matrices of each vertex.

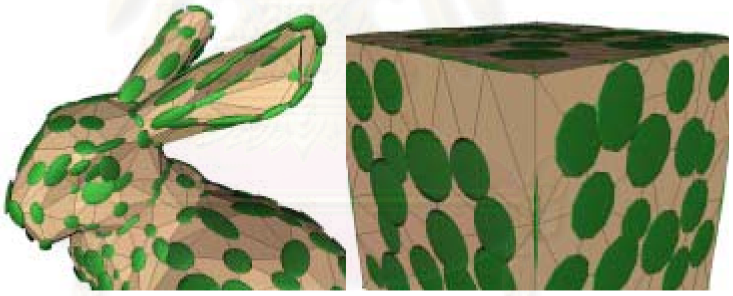


Figure 2-4: Examples of quadric isosurfaces

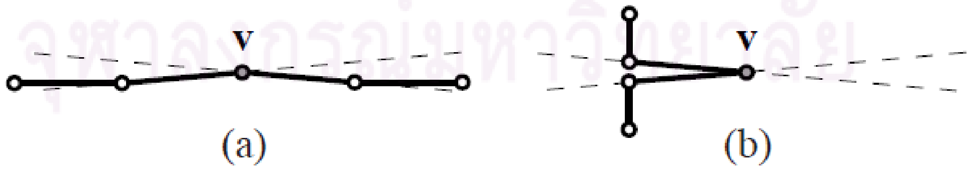


Figure 2-5: Ambiguity on sharp corners with QEM

This method has been noted to produce a good tradeoff between results of simplification and execution time compared to previous methods, and has been made widely available as QSlim. The main drawback of the regular QEM algorithm, however, is that the error is strictly geometric distance-based, and does not take other factors into account, such as sharp corners being interpreted the same as a near-flat surface (Figure 2-5). Garland and Heckbert have already proposed other additions to the method to improve mesh quality, such as allowing for contracting close vertices not directly connected by any edge, detecting and penalizing contractions that produce folded-over and/or narrow faces, and adding a quadric penalty to boundary vertices based on constraining planes, to help preserve boundary areas. Due to the ease of implementation and its availability, other papers have experimented with improving the QEM method, suggesting various forms of penalty, either by weighting vertices' matrices to favor the retention of certain vertices, applying penalties to the edge scoring, or expanding the matrices to include other properties in the calculation.

2.3.1.1 Quadric weighing

User-Guided Simplification (Kho and Garland, 2003) allows the user to weight the quadric matrices on features deemed visually important. Although it more or less ensures that the user will receive satisfactory results for simplifying a given model, it is not as useful or quick as fully-automatic simplification when multiple models need to be simplified.

Jong et al. (2006) calculate the torsion of each vertex's adjacent edges using approximations of the Frenet-Serret equations (Figure 2-6 shows the binormal vectors used to calculate torsion), and multiply the quadric matrices of each vertex with the distance-weighted average of the resulting torsion values for each adjacent edge. Their method makes the claim of improving the retention of features.

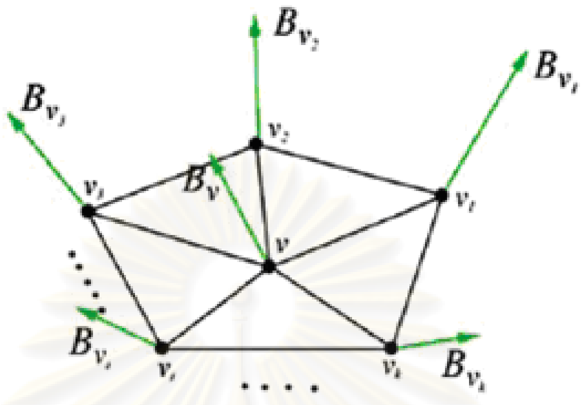


Figure 2-6: Binormal vectors for Jong's torsion detection method (2006)

Li and Zhu (2008) multiply each vertex's quadric matrix with a weighted sum of the average length of its incidental edges and the vertex's average "bending degree", based on the difference between the normal vectors of faces adjacent to the vertex. Figure 2-7 shows the determining of the bending degree.

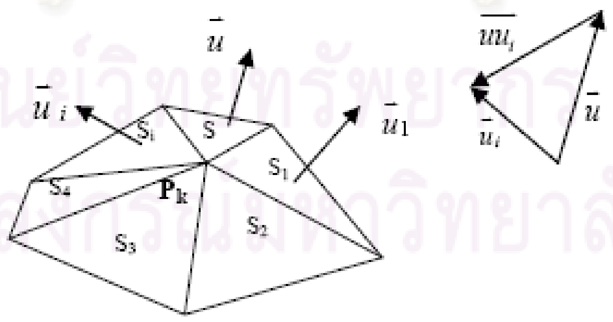


Figure 2-7: Determining bending degree in Li and Zhu's method (2008)

Tang et al. (2010) use the orientations of the faces adjacent to a given edge to calculate the average normal vector for the edge, and then use the angle between the average normal vector and each adjacent plane to calculate a “back cost” to weight the QEM matrix for the respective plane when calculating each vertex’s matrix (Figure 2-8). The algorithm always contracts any given edge to its midpoint.

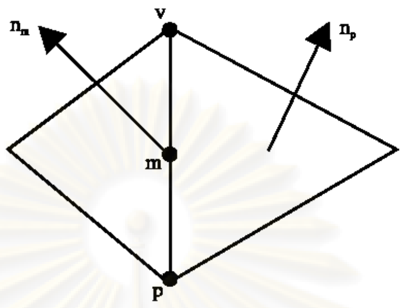


Figure 2-8: Back cost in Tang et al.’s method (2010)

2.3.1.2 Score penalizing

Similar to Li and Zhu’s approach, Xu et al. (2008) use the normal vectors of the vertex and its adjacent faces to calculate a feature value for each edge, which is then added to with the QEM score to assist in retaining features. In their implementation, the square root of the feature value, multiplied by a weighing factor, is added to one of the coefficients of the matrix; however, in effect, it is the same as adding the feature value multiplied by the square of weighing factor.

Kim et al. (2008) extend the QEM method to meshes with vertex color, by calculating color-related error separately from QEM geometric error, before multiplying the color-related error with the QEM error. Figure 2-9 shows the calculation of visual importance, a component of the color-related error.

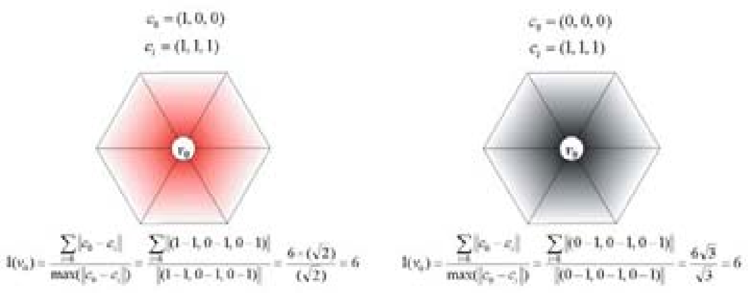

$c_0 = (1, 0, 0)$
$c_i = (1, 1, 1)$
v_0
$I(v_0) = \frac{\sum_{i=1}^{6} |c_0 - c_i|}{\max(|c_0 - c_i|)} = \frac{\sum_{i=1}^{6} |(1-1, 0-1, 0-1)|}{|(1-1, 0-1, 0-1)|} = \frac{6 \cdot (\sqrt{2})}{(\sqrt{2})} = 6$
$c_0 = (0, 0, 0)$
$c_i = (1, 1, 1)$
v_0
$I(v_0) = \frac{\sum_{i=1}^{6} |c_0 - c_i|}{\max(|c_0 - c_i|)} = \frac{\sum_{i=1}^{6} |(0-1, 0-1, 0-1)|}{|(0-1, 0-1, 0-1)|} = \frac{6\sqrt{3}}{\sqrt{3}} = 6$

Figure 2-9: Visual importance calculation from Kim et al. (2008)

Hussain (2009) calculates a vertex cost based on the normal field of each vertex's one-ring neighborhood for improved quality of simplification. His method calculates the normal field deviation over all edges adjacent to a given vertex, based on the areas and the normal vectors of its adjacent faces (Figure 2-10), and sums them to determine its cost, and then adds the normal field deviation for each given edge to its quadric error score to determine the cost of contracting each edge.

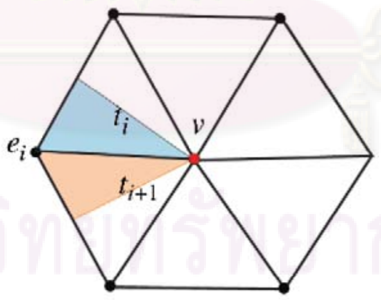


Figure 2-10: Normal field deviation over edge from Hussain (2009)

2.3.1.3 Expanded matrices

Garland and Heckbert's own extension to colored meshes (Garland and Heckbert, 1998) extends QEM to higher dimensions (up to 6), thus consuming more memory than with Kim et al.'s method, as the size of the matrix in higher dimensions is $O(n^2)$.

Wei and Lou (2010) extend QEM to higher-dimensions, using a 6-dimension feature sensitive matrix that combines QEM distance with normal variance. Figure 2-11 shows the isosurfaces of the matrices used in Wei and Lou's method. However, as with Garland and Heckbert's extension to colored meshes, the size of the matrix in higher dimensions is $O(n^2)$, thus increasing memory usage in a quadratic manner.

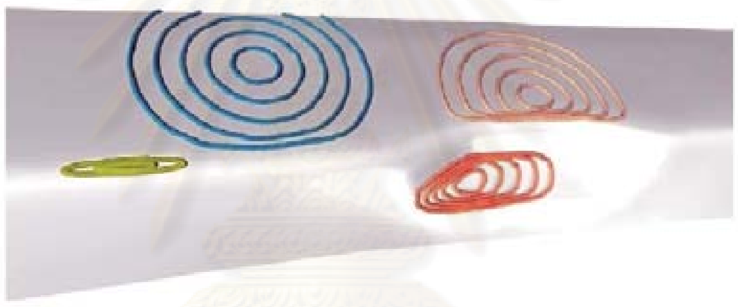


Figure 2-11: Feature sensitive isosurfaces in Wei and Lou's method (2010)

2.3.2 Non-QEM-based approaches

Although the QEM method among the most widely-used edge-contraction-based algorithms for mesh simplification, other non-QEM approaches have been proposed. These methods claim various strengths, such as the preservation of appearance, or using less memory.

2.3.2.1 Appearance-preserving

The Simplification Envelopes method (Cohen, 1996) calculates offset surfaces that limit the surface error (Figure 2-12), and then calculates surface intersections to constrain all contractions to remain within the envelopes, thus enforcing a geometric limit on surface error. However, this method requires complicated calculations for both the envelopes, and surface intersections.

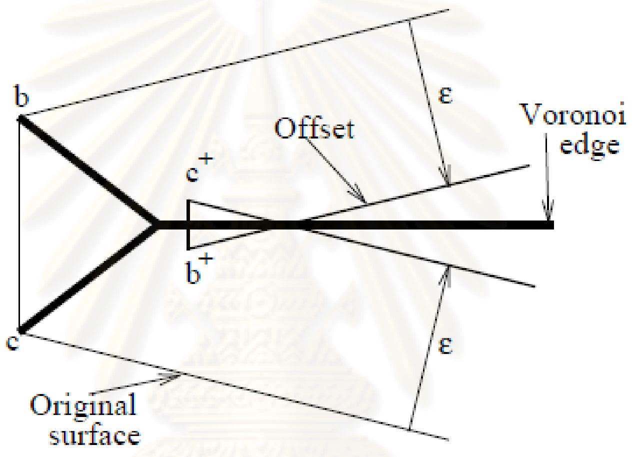


Figure 2-12: Determining offset surfaces for Cohen's method (1996)

Cohen et al.'s (1998) Appearance Preserving Simplification method uses parameterization of the model to normal and texture maps (Figure 2-13), along with a "texture deviation metric", to limit contractions to those that preserve the overall appearance of the model, by ensuring that the parameterized maps do not shift by more than a given error constraint. Using an error constraint allows for better quality in simplification, however, it also limits the level of simplification possible.

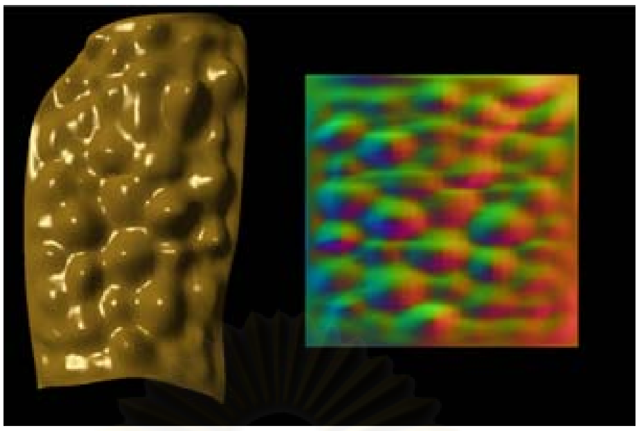


Figure 2-13: Parameterized normal map for Cohen et al.'s method (1998)

The Image-Driven Simplification (Lindstrom and Turk, 2000) method aims to preserve the appearance of the simplified model by rendering the model from various angles to calculate the error, thus causing the process to be is significantly slower. The use of rendering assures visual quality; however, although Lindstrom and Turk have proposed various techniques to reduce the rendering time, such as "re-rendering" only the affected portions of the image, purely geometric approaches are still faster than using direct rendering to determine the error.

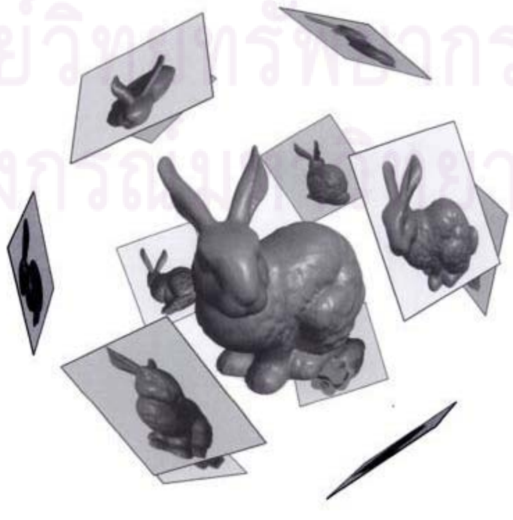


Figure 2-14: Rendering angles for Lindstrom and Turk's method (2000)

2.3.2.2 Memory-saving approaches

The Memoryless method (Lindstrom and Turk, 1998) saves on memory usage, by recalculating a quadratic error measure (different from Garland and Heckbert’s algorithm) for each affected vertex based on its current configuration, using volume and boundary preservation. Figure 2-15 shows a vertex being placed based on volume preservation and volume optimization constraints. This method has the cost of not storing the original structure for comparison and using extra running time. However, in practice, it has been shown to produce better geometric results than QEM.

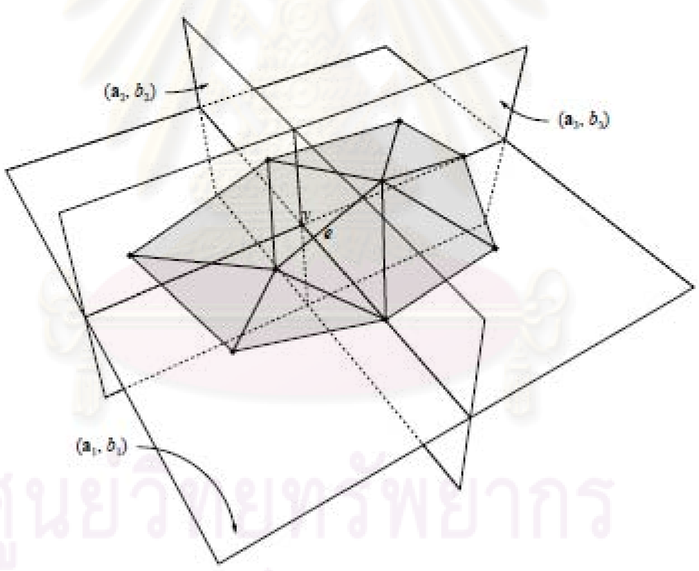


Figure 2-15: Determining the optimal vertex placement in the Memoryless method (1998)

Hussain et al. (2001) also take a memory saving approach, by using a product of swept area and angular deviation as the error of simplification. Their approach produces better running time than Lindstrom and Turk’s method, but is slightly slower than QSlim, and produces slightly worse Hausdorff distance results.

2.3.2.3 Subdivided meshes

Balmelli et al. (2002) have also proposed a method to simplify subdivided meshes with a 4-8 structure (that is, meshes in which the valence of vertices alternates between 4 and 8). It achieves this by pruning the hierarchy of vertex subdivisions, using a rate-distortion based metric, thus allowing for a “general decimation” method that removes many vertices per cycle. Figure 2-16 shows their result of pruning the subdivision hierarchy. The authors suggest that the method provides an *O*(*n* log *n*) optimal mesh simplification algorithm for the case of 4-8 structures (in contrast to NP-Hard for the general case).

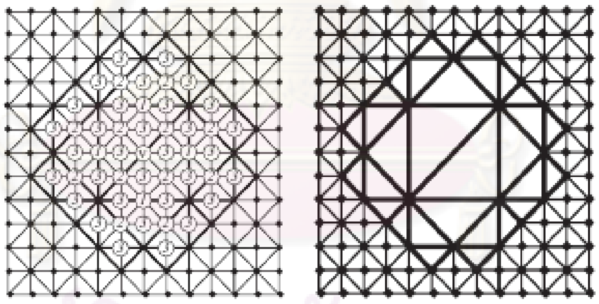


Figure 2-16: Pruning the hierarchy in Balmelli’s method (2002)

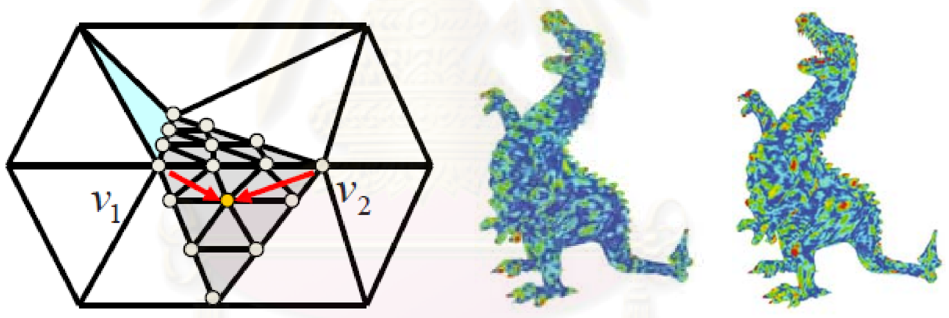
2.3.2.4 Global properties

Tang et al. (2007) propose two different and complex moment-based metrics based on global feature change, one based on the surface, the other based on volume. Their claim is that these metrics take global change into account. The results show that this approach produces better volume and surface moment results than

QEM; however, they do not use Hausdorff distance results as part of their result evaluation.

2.3.2.5 Optimal placement

Choi et al. (2008) uses shape compactness, angular deviation and curvedness as factors to determine the optimal position from contracting each edge, based on a set of points subdivided from a Bézier patch around the edge. This method is different from the normal procedure of determining the position with the least error cost. Although it produces similar average surface error to QEM, it produces more areas of high surface error (in Figure 2-17(b) and (c), red spots have high surface error).



(a) (b) (c)

Figure 2-17: Selecting vertex from subdivided Bézier patch in Choi’s method (a), results from dragon model using QEM (b) and Choi’s method (c) (2008)

2.3.3 Heap updating process

Another topic concerning the mesh simplification process is how to update the priority heap. Most edge-contraction based methods are assumed to perform a full heap update after each contraction, that is, each entry in the heap is checked before updating if necessary. However, a handful of papers have addressed the issue of saving time on this step:

Cohen et al. (1997) defer the updates of edge scores by assigning a “dirty flag” to scores that require updating, and re-calculate and re-insert flagged scores when they are encountered in the heap. A possible pitfall for this method is the possibility of encountering many consecutive “dirty flagged” edges, thus limiting progress.

Wu and Kobbelt (2004) avoid using a heap altogether with their “multiple-choice” method, by randomly picking a small subset of edges, calculating each edge’s score, and contracting the edge with the least score. Their reasoning is that when models are simplified at a drastic level, most edges are likely to be contracted anyway.

Chen et al. (2004) save on heap space and execution time by filling the heap up only to a given size limit, and then alternate performing contractions and calculating scores for heap insertion, so as to keep the size of the heap under the limit. Their research claims a constant time for each contraction step (and thus linear time for the overall process).

2.4 Our observations

For most practical mesh reduction methods, we believe that the process should, in addition to producing well-simplified models, work generally well over a wide range of meshes, perform the simplification without requiring user intervention during the process, rely entirely on geometry-based factors to determine score, and rely on factors that are easily calculated, for speed and coding purposes.

More importantly, however, we have observed that mesh simplification methods that use curvature to improve on QEM generally only use a single score per vertex, which may result in ambiguity with surfaces with different properties receiving the same score. The curvature of a surface can generally be defined using two principal normal curvatures, the maximum ($k_{\max}$) and minimum ($k_{\min}$) curvatures of the local area around a given point, and the directions of the curvatures. We believe that using both curvatures along with their directions can help provide better scoring.

We have also observed that most methods assume a full update on the priority heap after each contraction. Some research has been published that suggests that full updates after each contraction are not necessary for producing quality results from simplification. However, we feel that there should be some degree of assurance that an edge with a “good” score is being contracted at each step (instead of randomly picking a set of edges and expecting the best score among them to be “good”), and that progress is constantly being made in the simplification process (instead of having to repeatedly recalculate scores due to the “dirty flag”).

2.5 Summary

In this chapter, we have discussed background knowledge about polygonal meshes and data structures used to store them, mesh simplification and previous edge-contraction based mesh simplification methods. Polygonal meshes are the most popular method for representing three-dimensional models, using vertices, edges and faces to represent the surface. Many data structures exist to store this information, such as winged-edge and cell lists. Most mesh simplification algorithms do not specify a specific structure, although a handful exist that do. Mesh simplification can be used to lower the number of faces in a complex three-dimensional model. The simplified models can then be more easily stored or rendered using typical hardware. Another reason to simplify models is to reduce the detail of objects far from the viewer.

The problem of optimal mesh simplification is NP-Hard, so research on mesh simplification methods focus on creating heuristics to produce a good result. Many classifications for mesh simplification algorithms exist: mechanism, topology handling and type. The most researched mechanism is edge contraction, and the preferred approach is dynamic simplification, as these properties lend well to level-of-detail based structures, which allow for models to be represented with an exact number of faces. Based on the state of the art of edge-contraction based algorithms, we classify the works into two categories: QEM-based and non-QEM-based. Garland and Heckbert's QEM algorithm uses the sums of the distances from the planes of adjacent faces to calculate error score. Since the error is distance-based, it has potential problems, such as ambiguity at sharp corners. QEM-based algorithms take QEM algorithm and add other factors to the score calculation to improve handling of features, while non-QEM-based algorithms calculate scores based on factors such as volume optimization, surface and volume moment, or use normal maps, offset surfaces, or direct rendering to control the process. Also, we note that a handful of papers focus on how to save time updating the priority heap used in edge-contraction-based algorithms.

From our observations, we note that QEM-based algorithms only use a single score per vertex to assist in score calculation. We hypothesize that using both principal curvatures of a surface can help improve the performance. We also consider that heap updating algorithms should guarantee good scores and constant progress.

CHAPTER III

THE SIMPLIFICATION ALGORITHM

In this chapter, we will explain the proposed simplification algorithm. We begin by describing the general overview for the algorithm, along with our notational conventions and the scope of the models that we use. We then describe each step in detail, explaining our implementation choices, before providing a time and memory complexity analysis.

3.1 Overview

The algorithm will follow a similar approach to a typical edge-contraction based simplification algorithm (Figure 3-1): After reading the input data, we then calculate the score of each edge and place the scores in a priority queue. We then contract edges with the best score, and update the scores in the queue, and then repeat the contraction and update processes until the desired level of detail (LOD) has been reached, or the priority queue is empty. We will explain our notation and scope, before explaining each of the steps in detail.

```
Algorithm: Proposed Algorithm
Input: Polygonal Mesh (P)
Output: Simplified Mesh (P')
    1. Convert P to Abstract Cellular Complex
    2. Score calculation,
    3. Repeat {
    4.         Edge contraction
    5.         Heap update
    6. } Until desired LOD reached
    7. Output P'
    8. End
```

Figure 3-1: Algorithm for edge-contraction simplification

3.1.1 Notation

In the following explanation of our algorithm, we will use the following notations and definitions:

v_x	Vertex (Figure 3-2)
e_x or $<v_x, v_y>$	Edge (Figure 3-2)
f_x or $<v_x, v_y, v_z>$	Face (Figure 3-2)
$F(v_x)$	Set of faces adjacent to vertex v_x (Figure 3-2)
$S(v_x)$	Vertex to which v_x is currently mapped
$S^{-1}(v_x)$	Set of vertices that currently map to v_x (i.e., the inverse of S)
$U(v_x)$	Update cycle at which v_x was most recently affected
$U(e_x)$	Update cycle at which the score for contracting edge e_x was most recently updated

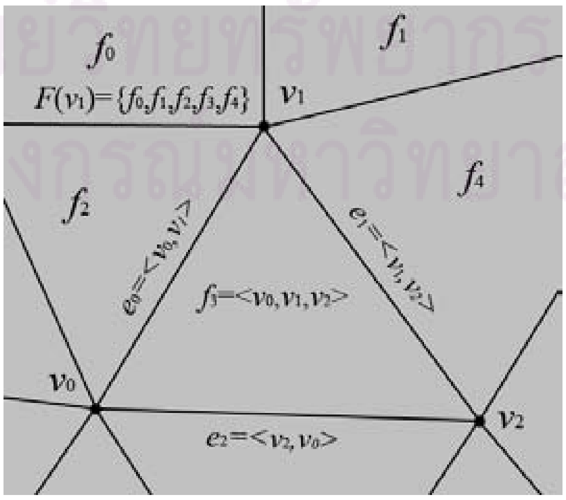


Figure 3-2: Explanation for v_x, e_x, f_x, $<v_x, v_y>$, $<v_x, v_y, v_z>$, and $F(v_x)$

3.1.2 Scope

For our algorithm, we use as our input polygonal meshes with the following properties:

- singular objects
- no textures or colors
- triangular faces
- mostly manifold topology (i.e., boundaries are allowed, as well as a small amount of non-manifold edges with more than two faces adjacent)

3.2 Converting to Abstract Cellular Complex

Converting a face-vertex based polygonal mesh to Abstract Cellular Complex format proposed by Kovalevsky (2001) is rather trivial. After inputting the faces and vertices, for each face, we determine its edges by storing each pair of consecutive endpoints in a binary tree, with lower numbered vertex first, to prevent duplicate edges from being stored. After obtaining all the edges, we then determine and store the faces adjacent to each edge, along with the edges adjacent to each vertex. As the structure calls for specifying a start- and end-point for each edge, we simply designate the lower-numbered vertex as the start-point.

The Abstract Cellular Complex format also allows for volumes to be represented, as well as the right hand rotation order of faces around a given edge;

however, as we do not use these data in our algorithm, we do not need to determine this information.

3.3 Score calculation

After receiving the mesh data input, we first determine the faces adjacent to a given vertex, boundary edges (i.e., edges with exactly one adjacent face), and boundary vertices (the vertices of the boundary edges). Using the cell-list structure, these topological relationships can be determined at relatively small computational cost.

We also calculate the principal curvatures and their directions for each vertex, by calculating the normal vectors at each vertex, and then using Batagelo and Wu's method (Batagelo and Wu, 2007) of using linear least squares to estimate the curvature tensor, and then calculating the eigenvalues and eigenvectors from the resulting tensor. We have chosen Batagelo and Wu's method, because their method is easily implemented, and it is claimed to be fast, as well as robust on noisy meshes, producing fewer outliers than other methods.

To calculate the overall score, we first calculate the quadric error based on Garland and Heckbert's method (the QEM score), and then calculate the regularity and angular deviations of the resulting faces to penalize the score. We will now describe each part of the process in detail.

3.3.1 QEM Score

After we have calculated the curvatures, we next calculate the quadric matrices of each vertex. For each face, we calculate its normal vector to determine its corresponding plane, and convert the plane to a K_p matrix as per the regular QEM method. To each of the face's constituent vertices, we weight it with the product of the area and incident angle on the vertex. After inspecting every face, we perform a weighted average to obtain a quadric matrix $Q(v)$ for each vertex:

$$Q(v) = \frac{\sum_{i=1}^{k} K_{p_i} \theta_i w_i}{\sum_{i=1}^{k} \theta_i w_i} \tag{3-1}$$

Where θ_i is the angle of face f_i with area w_i incident on v (see Figure 3-3). It should be noted that this is the similar to one of the many possible ways Garland and Heckbert propose to weight the quadric matrices to reduce the effects of tessellation, except that Garland and Heckbert do not average the matrices, in order to retain scale variance.

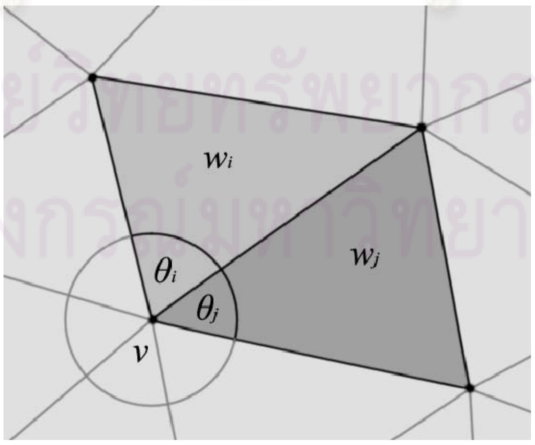


Figure 3-3: Explanation for angular weighting method

After we have prepared the data for simplification, we then determine, for each edge $e = <v_x, v_y>$, whether e is suitable for contraction. First, we check the valences of the vertices immediately adjacent to both endpoints, discarding any edges where any such vertex has a valence of 3. This is because contracting the edge will result in two of the faces surrounding the edge having the same vertices, resulting in the resulting edge effectively becoming adjacent to three faces.

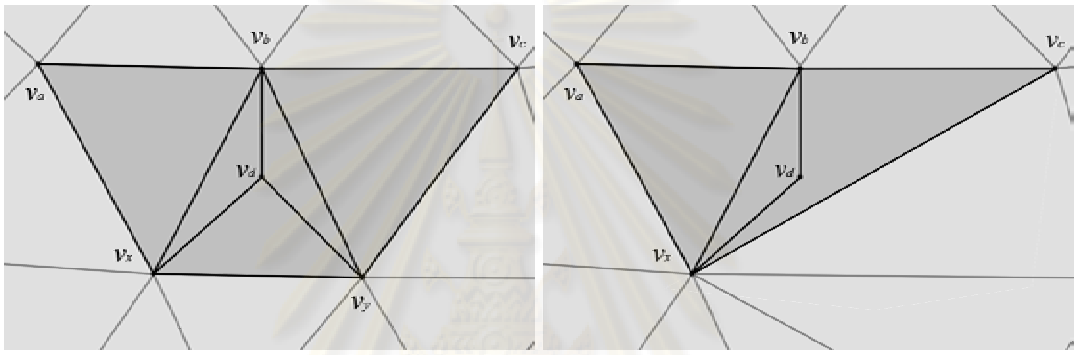


Figure 3-4: Edge contraction in the area around a vertex with a valence of 3

To explain, in Figure 3-4, v_d has three faces around it: $<v_x, v_b, v_d>$, $<v_y, v_b, v_d>$, and $<v_x, v_y, v_d>$. After contracting $<v_x, v_y>$ to v_x, the first two faces are effectively combined into a single face, while the third face becomes degenerated. This results in $<v_x, v_b>$ being adjacent to three faces: $<v_a, v_b, v_x>$, $<v_x, v_b, v_d>$ (representing both original and the former $<v_y, v_b, v_d>$), and $<v_x, v_b, v_c>$ (formerly $<v_y, v_b, v_c>$), whereas the pre-contraction configuration has no such edges.

We also determine the number of vertices that are adjacent to both endpoints, and compare to the number of faces adjacent to the edge, and discard edges where these two values are not equal. This prevents a contraction from altering the topology of the model, such as by closing up a triangular hole (Figure 3-5), or closing up a "cylinder" with a triangular opening (Figure 3-6): In Figure 3-5, two vertices are adjacent to both v_x and v_y, but only one face is adjacent to $<v_x, v_y>$, as the edge is opposite a hole. Contracting $<v_x, v_y>$ closes up the hole. In Figure 3-6, two faces are adjacent to both v_x and v_y (highlighted on the left), but three vertices are adjacent due to a triangular opening in the structure (v_x, v_y, v_c). Contracting $<v_x, v_y>$ to v_x results in the opening being closed up, and the edge $<v_x, v_c>$ becoming adjacent to four faces.

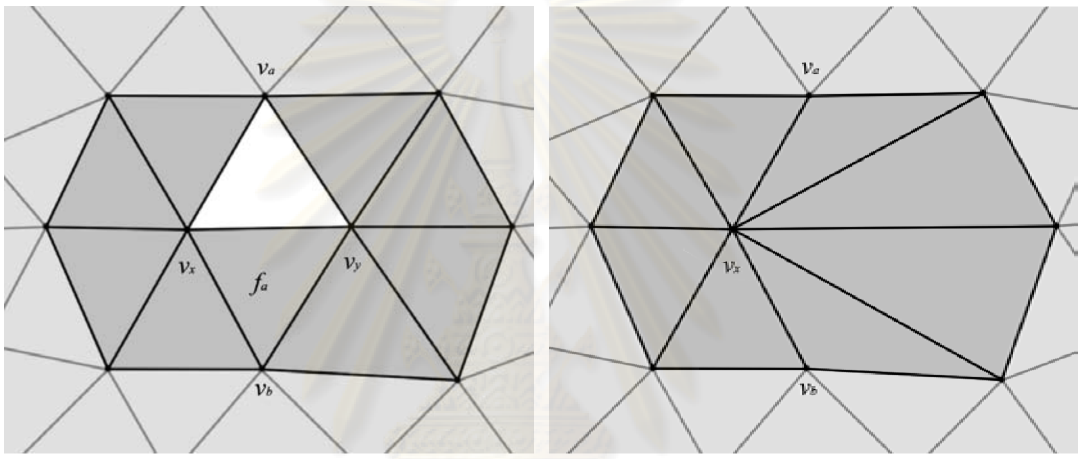


Figure 3-5: Contracting triangular hole

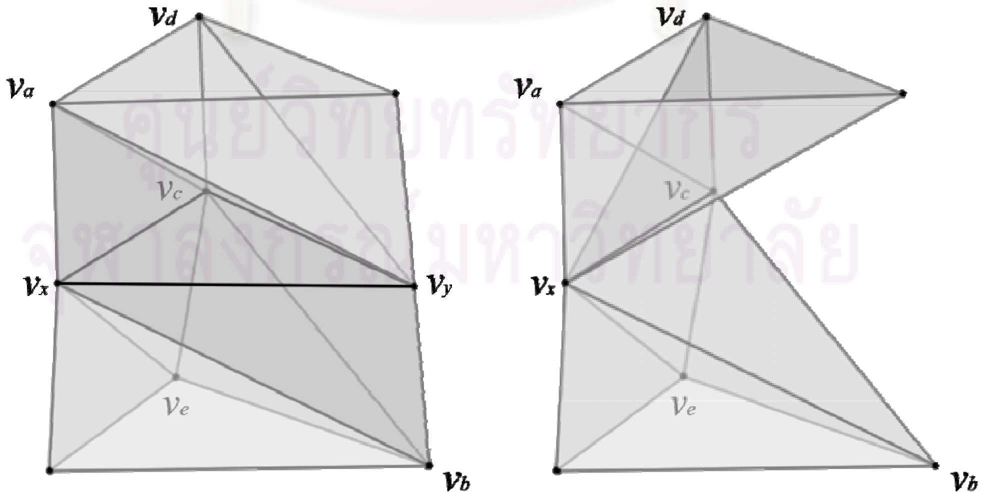


Figure 3-6: Contracting around triangular opening in model

If the edge passes this test, we then determine how to contract each edge of the model, and the score of the contraction. There are two possible choices for determining the position of the vertex resulting from edge contraction: Determine the position with the minimal error score or choosing the endpoint with the lesser score (also known as subset selection or half-edge contraction). Although the former allows for lesser error, choosing between endpoints allows for easier calculation and also avoids having to store coordinate differences in level-of-detail based data structures; therefore, we have chosen the latter method.

We first calculate the score of contracting the edge to either of its endpoints. We first calculate the QEM score in the same fashion as the regular QEM method, by summing together the matrices of the edge's endpoints and using the summed matrices to calculate the QEM score. We then determine the absolute normal curvature from the contracted point to the result point; that is, when contracting v_x to v_y, we determine the normal curvature in the direction from v_x to v_y. Where $k_{\max}$ and $k_{\min}$ are the maximum and minimum principal curvatures of v_x respectively, $\vec{N}_{v_x}$ is the normal vector of v_x, and $\vec{D}$ is the direction for $k_{\max}$, the absolute normal curvature K from v_x to v_y can be calculated as follows:

$$\vec{D}' = \left(\vec{N}_{v_x} \times v_x v_y\right) \times \vec{N}_{v_x} / \left\| \left(\vec{N}_{v_x} \times v_x v_y\right) \times \vec{N}_{v_x} \right\| \qquad (3\text{-}2)$$

$$\cos^2\theta = (\vec{D} \cdot \vec{D}')^2 \qquad (3\text{-}3)$$

$$K = \left| \cos^2\theta k_{\max} + (1 - \cos^2\theta) k_{\min} \right| \qquad (3\text{-}4)$$

Where $\vec{D}'$ is the unit vector representing the direction from v_x to v_y projected on to the plane defined by $\vec{N}_{v_x}$, and θ is the angle between $\vec{D}'$ and $\vec{D}$. We multiply the score with the absolute normal curvature along with the edge length, as the contraction of a longer edge is more likely to affect the shape of a model than a shorter edge. The QEM score obtained for contracting <v_x, v_y> to v_x is

$$s'(\langle v_x, v_y \rangle, v_x) = \|v_x v_y\| K v_x^{\mathsf{T}} \left(Q(v_x) + Q(v_y) \right) v_x \tag{3-5}$$

3.3.2 Penalties

After obtaining the QEM score, we then apply penalties based on various properties, starting with facial regularity or compactness, which is desirable for applications such as finite element analysis. We calculate the regularity γ of each resulting face as per Guèziec (1995):

$$\gamma = \frac{4\sqrt{3}w}{l_1^2 + l_2^2 + l_3^2} \tag{3-6}$$

Where w is the area of the face, and the l_i are the lengths of its edges. This equation produces a result of 1 for an equilateral triangle, and 0 for a degenerate triangle. We determine the face with the least regularity $\gamma_{\min}$, and penalize contractions that result in faces with lower than 0.5 thusly:

$$p_{reg} = \begin{matrix} \left(\frac{0.5}{r_{\min}} \right)^2 - 1 & r_{\min} < 0.5 \\ 0 & r_{\min} \geq 0.5 \end{matrix} \tag{3-7}$$

Our choice of 0.5 allows for a degree of variance in facial regularity from being perfectly equilateral without any penalty added to the contraction score. Also, the above equation results in a score that tends towards positive infinity as the regularity decreases to 0.

The next penalty we apply is related to facial orientation. We determine the orientation of the resulting faces using a cross product, and compare the angle θ with the face's original orientation using a dot product, before penalizing according to the largest angle of change in orientation $\theta_{\max}$ thusly:

$$p_{ang} = \begin{cases} \dfrac{\sin\theta_{\max}}{1-\sin\theta_{\max}} & \theta_{\max} < 90^{\circ} \\ +\infty & \theta_{\max} \geq 90^{\circ} \end{cases} \tag{3-8}$$

Our choice of 90 degrees as the cutoff limit ensures that faces will never flip over in relation to their original orientations. This procedure is also easier to calculate than Garland and Heckbert requiring the resulting vertex to fall within a given area.

We also compare the angle of each face's orientation to the normal vectors of the resulting vertices and calculate a curvedness-inverse-weighted average θ', where curvedness is a Pythagorean sum of the maximum and minimum curvatures ($R = \sqrt{k_{\min}^2 + k_{\max}^2} / \sqrt{2}$), under the reasoning that normal vectors at areas of high curvedness are less representative of the ideal facial orientation than those at areas of lower curvedness. We then apply the same penalizing method as above. Where R_{v_i} is the curvedness of v_i and $\angle(\vec{N}_f, \vec{N}_{v_i})$ is the angle between the normal vector of f and the normal vector of v_i:

$$\theta' = \frac{\sum \dfrac{\angle(\vec{N}_f, \vec{N}_{v_i})}{R_{v_i}}}{\sum \dfrac{1}{R_{v_i}}} \tag{3-9}$$

$$p_{avg} = \begin{cases} \dfrac{\sin\theta'_{\max}}{1-\sin\theta'_{\max}} & \theta'_{\max} < 90^{\circ} \\ +\infty & \theta'_{\max} \geq 90^{\circ} \end{cases} \tag{3-10}$$

We also consider the dihedral angles between the affected faces, and faces immediately adjacent to any affected faces, and compare them to the dihedral angles of the original orientations (Figure 3-7). Note that during later stages of the simplification, the faces being compared may not have been adjacent to each other in the original model.

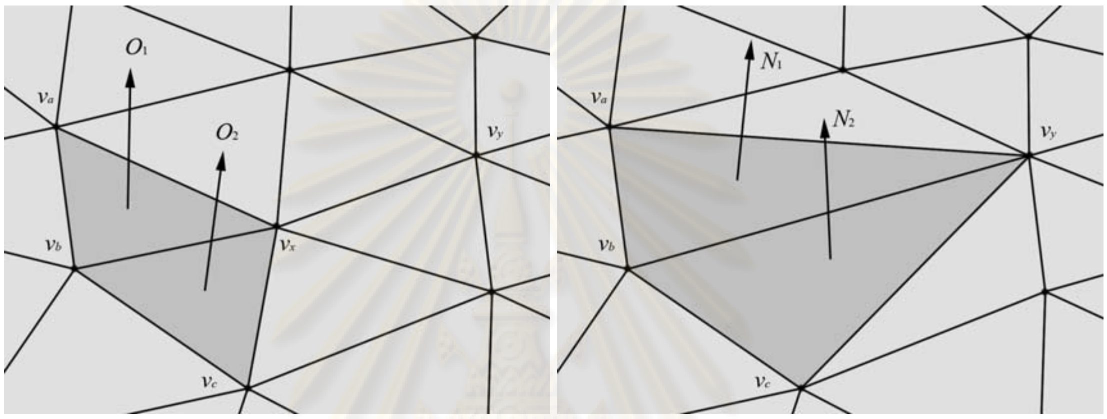


Figure 3-7: The change in dihedral angle between two faces after contracting <v_x, v_y> to v_y

We also consider that the relative orientation of each pair of faces may have changed from concave to convex, or vice versa. We consider that such changes in orientation are less desirable; therefore, in our implementation, we detect such a result, and apply an extra penalty to such contractions:

$$\hat{\theta}(\vec{O}_1, \vec{O}_2, \vec{N}_1, \vec{N}_2) = \begin{cases} \left|\sin^{-1}\left(\left\|\vec{N}_1 \times \vec{N}_2\right\|\right) - \sin^{-1}\left(\left\|\vec{O}_1 \times \vec{O}_2\right\|\right)\right| & (\vec{N}_1 \times \vec{N}_2)\cdot(\vec{O}_1 \times \vec{O}_2) > 0 \\ 2\sin^{-1}\left(\left\|\vec{N}_1 \times \vec{N}_2\right\|\right) + \sin^{-1}\left(\left\|\vec{O}_1 \times \vec{O}_2\right\|\right) & (\vec{N}_1 \times \vec{N}_2)\cdot(\vec{O}_1 \times \vec{O}_2) \le 0 \end{cases} \tag{3-11}$$

Where $\vec{O}_1$ and $\vec{O}_2$ are the original unit normal vectors of two faces, and $\vec{N}_1$ and $\vec{N}_2$ are the unit normal vectors of the same faces after the contraction. We then use the highest dihedral angle score to calculate the penalty, in the same way as the previous angles above:

$$p_{rdc} = \begin{cases} \dfrac{\sin\hat{\theta}_{\max}}{1-\sin\hat{\theta}_{\max}} & \hat{\theta}_{\max} < 90^\circ \\ +\infty & \hat{\theta}_{\max} \geq 90^\circ \end{cases} \tag{3-12}$$

Next, we take boundaries into consideration, by disallowing any contractions that contract a boundary vertex to a non-boundary vertex, while allowing contractions in the opposite direction. This ensures that the vertices of each boundary in the simplified model are a subset of those in the original model's corresponding boundary. We only allow boundary vertices to be contracted if the contracted edge is part of the boundary, and we also apply an extra penalty to the contraction based on overall change in boundary area.

We determine the change in the boundary area thusly: When calculating the score for the contraction of a boundary edge, we determine the boundary vertex that would become adjacent to the contracted vertex, and calculate the area of a triangle between that vertex and the edge to be contracted, which is the area would be immediately affected by the contraction. For example, in Figure 3-8, when contracting <v_x, v_y> to v_y, we determine that v_z would then become adjacent to v_y, resulting in the area <v_x, v_y, v_z> being affected.

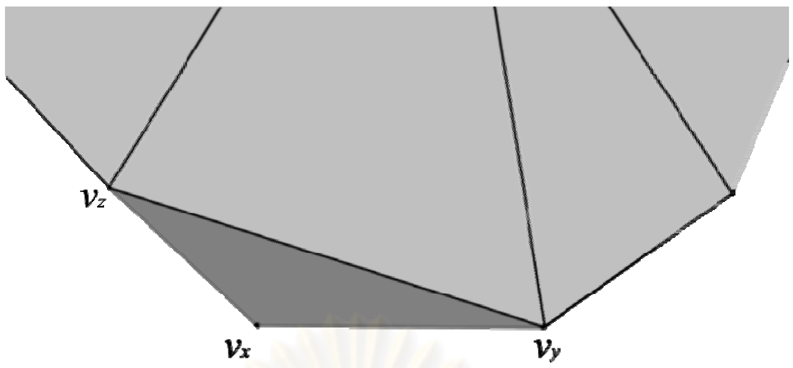


Figure 3-8: Boundary handling during initialization

After we have obtained the affected area, we calculate its ratio ρ to that of an equilateral triangle with the same edge length as the remaining edge. From Figure 3-8, the ratio would be calculated thusly:

$$\rho = \frac{A\left(\left\langle v_x, v_y, v_z \right\rangle\right)}{\left(\sqrt{3}/4\right)\left\|\left\langle v_y, v_z \right\rangle\right\|^2} \tag{3-13}$$

Where $A\left(\left\langle v_x, v_y, v_z \right\rangle\right)$ is the area of a triangle with $<v_x, v_y, v_z>$ as its corners. We then use the ratio to calculate a score for the boundary change, similar to the angular scores, to prevent severe changes in the boundary:

$$p_{bdr} = \begin{matrix} \dfrac{\rho}{1-\rho} & \rho < 1 \\ +\infty & \rho \geq 1 \end{matrix} \tag{3-14}$$

The boundary checking process is optional, and may be skipped when simplifying models that have few or no boundaries. (Our implementation automatically skips the scoring portion of the process when the boundary score has been assigned a weight of 0; however, it still prevents the contraction of boundary vertices to non-boundary vertices.) Where $s'\left(\left\langle v_x, v_y \right\rangle, v_x\right)$ is the original QEM score from contracting <v_x, v_y> to v_x, and α, β, and δ represent user-defined weights for each part of the score, the final error score we obtain is as follows:

$$s(\langle v_x, v_y \rangle, v_x) = \log(s'(\langle v_x, v_y \rangle, v_x)) + \alpha(p_{ang} + p_{avg} + p_{rdc}) + \beta p_{bdr} + \delta p_{reg} \qquad (3\text{-}15)$$

Taking the logarithm of the QEM score before adding the penalties effectively multiplies the penalties (to the power of the user-defined weights) with the QEM score. However, we will also experiment with a linear combination addition-based penalizing method. After we have obtained the final scores of either possible contraction of each edge, we insert the better score into a priority heap H, along with the following information: the two endpoints (with the better-scoring endpoint listed first), the original edge e, and the update cycle when the score was most recently updated $U(e)$, initially −1, for the purpose of updating the scores. In our implementation, we have implemented H as a standard array with a value declaring its size $|H|$.

3.4 Edge contraction

As in most other edge-contraction based algorithms, we obtain the edge with the best score from the top of H and contract it, removing faces adjacent to the contracted edge. While contracting edges, we keep track of the faces adjacent to each vertex using F for the purpose of score calculation. We also keep track of vertex mappings using S and S^{-1}, and when each vertex has been affected by previous contractions, for the purposes of score updating, using U. Initially, for every vertex v, $S(v)=v$, $S^{-1}(v)=\{v\}$, and $U(v)= -1$.

For point-face relationships in F: When contracting <v_x, v_y> to v_x, the faces adjacent to exactly one of the endpoints become the faces adjacent to the remaining vertex. Faces adjacent to both endpoints (i.e., adjacent to the edge) become degenerate and can be removed. Therefore:

$$F(v_x) \leftarrow F(v_x) \cup F(v_y) - (F(v_x) \cap F(v_y)) \qquad (3\text{-}16)$$

For example, in Figure 3-9, the edge $<v_x, v_y>$ has been contracted to v_x. The faces adjacent to either v_x or v_y (light and medium shading) have now become adjacent to v_x, while faces adjacent to the edge (dark shading) have become degenerated.

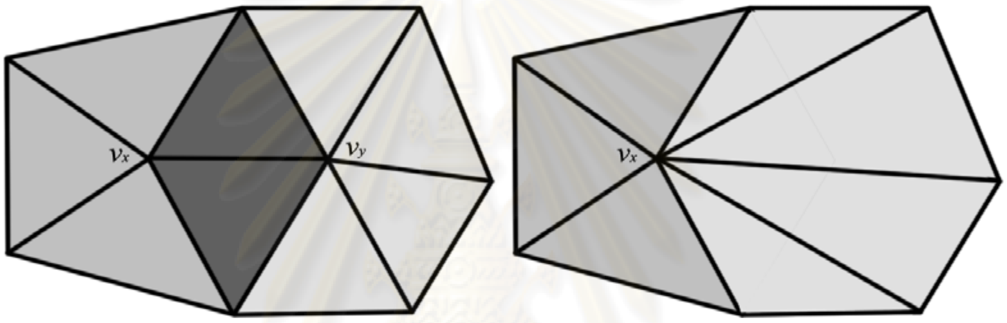


Figure 3-9: Point-face relationships before and after edge contraction

Using S and S^{-1} to track vertex mappings allows us to convert any original face of the model to its face in the model at the current level of simplification (and thus determine a face's validity). Storing mappings in both directions in our implementation also allows the mappings to be quickly updated: without the inverse sets, updating would require making a pass over all vertices to check for and update mappings, while the inverse indicates exactly which vertices to update. When contracting $<v_x, v_y>$ to v_x, vertices that were originally mapped to either vertex are now mapped to v_x, that is, the resulting inverse is the union of the two sets:

$$S^{-1}(v_x) \leftarrow S^{-1}(v_x) \cup S^{-1}(v_y) \qquad (3\text{-}17)$$
$$\forall v \in S^{-1}(v_x) \cup S^{-1}(v_y), S(v) \leftarrow v_x \qquad (3\text{-}18)$$

To save on memory usage, we remove the QEM matrices and S^{-1} and F sets from contracted vertices immediately after contraction, by setting them to null references. It can be easily shown that, using this method, the total memory used to store S^{-1} remains constant throughout the execution, while the total memory for F reduces according to the number of remaining valid faces, as in both cases, each vertex (or valid face) is represented exactly once (or 3 times) in total amongst the sets of S^{-1} (or F).

3.5 Heap updates

The next step in mesh simplification using an edge-contraction mechanism is to update the priority heap. Most methods are assumed to perform full updates after each contraction. However, we have chosen to perform updates differently from typical edge-contraction based algorithms, by using a novel partial and lazy updating scheme. We describe our scheme in detail, and explain our use of affected vertices to determine which edges to update.

3.5.1 Partial updating scheme

Our scheme aims to combine the concepts of Cohen et al.'s "dirty flag" and Wu and Kobbelt's "multiple-choice" methods as described in the Previous Work section, while providing both a better assurance of contracting a good score and making constant progress during the process, by only updating upon finding an outdated score, and then updating only the topmost portion of H. Our reasoning is that scores that are near the top of H will likely not be affected by contractions, and will thus remain close to the top after the update.

The details of our scheme are as follows:

- Check whether top score needs to be updated, based on vertex mappings and update information, or a certain number of edges have been contracted since the last update. If not, perform contraction, else perform update.

- For all contractions in the top $n-5$ levels of the heap, where $n = \lfloor \log_2 |H| \rfloor$, re-calculate score if necessary to update. Our choice of $n-5$ provides a balance between updating fewer entries resulting in more frequent updates, and updating more entries than necessary. It can be shown that the number of entries in the top $n-5$ levels is between $|H|/32$ and $|H|/64$. We will also experiment with varying the number of updated levels.

- Restore heap property by swapping values, starting from the last updated score upwards.

3.5.2 Affected vertices

To determine which scores need to be updated, we take into account which vertices have been affected and when. Each edge contraction affects the score of several other contractions involving vertices in the nearby area. Any triangles adjacent to the contracted vertex will change their shape, thus requiring any scores involving those faces to be recalculated. Also, any edges with either vertex are also affected, as the contracted vertex now maps to a new vertex, while the remaining vertex has new faces adjacent to it, as well as a new quadric matrix (the sum of the original vertices' matrices). Therefore, when contracting $<v_x, v_y>$ to v_x:

- For every triangle T in $F(v_y)$, all vertices in T are affected,
- For every triangle T adjacent to any triangle in $F(v_y)$, all vertices in T are affected, and
- v_x is affected

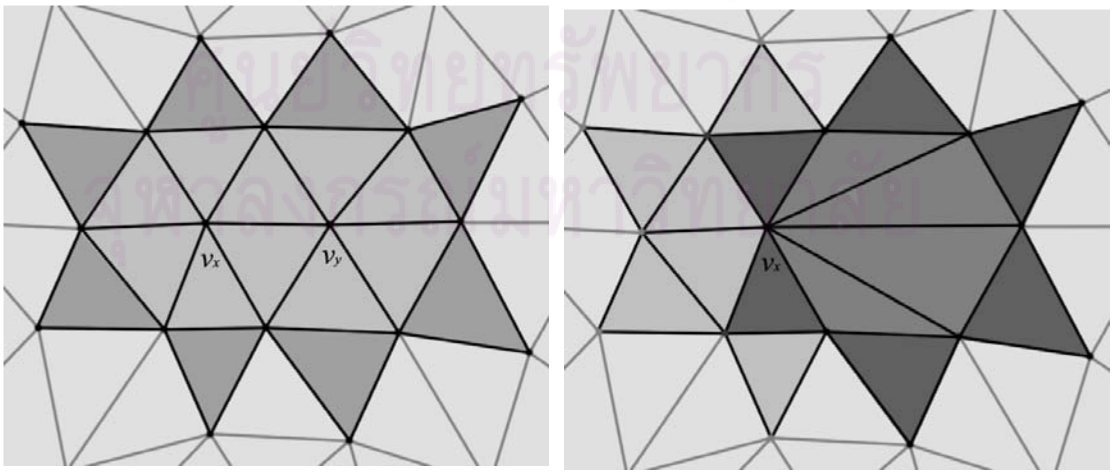


Figure 3-10: Illustrating how vertices are affected by a contraction

To justify this reasoning, Figure 3-10 shows the effects of a contraction from $<v_x, v_y>$ to v_x. In the left figure, the shaded faces are the faces considered when calculating the score of the edge, namely: faces adjacent to either of the two vertices (lightly shaded), or triangles that share an edge with any such face (medium shaded, for determining the dihedral angle score). In the right figure, the shaded faces (originally adjacent to v_y) have been altered due to the contraction; therefore, any vertices adjacent to any of those faces are affected. Also, the darker shaded faces are adjacent to the affected faces, resulting in a change in dihedral angle; hence, the vertices of those faces are also affected. The affected vertices are marked in black in the right figure, while unaffected vertices are in gray. v_y is also considered to be affected, as any edges that originally had v_y as an endpoint will now have v_x instead.

To assist in tracking when each vertex has been most recently affected, we store a value U representing the current update cycle, incrementing it at each update. Our storage of $U(v)$ and $U(e)$ allows us to compare when the score was most recently calculated, with when either endpoint was last affected, to determine whether the score needs to be updated.

When we mark vertices as being affected, we store the current value of the update cycle plus one: $U(v_x) \leftarrow U+1$. This is so that if two contractions involving the same vertex are encountered during the same update cycle, only the first update will be performed, and the second will trigger a new update cycle. Also, when updating a score immediately after either of its vertices has been affected, the score's update cycle will be equal to that of the affected vertex. Similarly, we also keep track of when each face has been affected through contractions, under the simpler condition that when contracting v_x to v_y, all faces in $F(v_x)$ are affected.

The criteria for when an edge's score needs to be updated are as follows:

- Either endpoint of the edge now maps to a different vertex (checked using vertex mappings): $S(v_x) \neq v_x$ or $S(v_y) \neq v_y$
- The most recent update for either endpoint is more recent than that of the score: $\max(U(v_x), U(v_y)) > U(e_x)$

One could also inspect whether the most recent update for any face in $F(v_x)$ or $F(v_y)$, or any faces adjacent to such faces, is more recent than that of the score, however, the endpoint update condition is easier to check, especially when considering the faces adjacent to those in $F(v_x)$ or $F(v_y)$. We also choose to automatically invoke an update after a certain number of edges have been contracted without any updates being invoked otherwise. We have chosen this limit to be 100, but we have observed that this limit does not appear to significantly affect the overall execution time.

3.5.3 Score re-calculation

When we re-calculate the score for a given edge, we determine its endpoints using the vertex mappings, and check that the endpoints are different, before determining whether score re-calculation is necessary based on the vertex mapping, $U(v)$ and $U(e)$, and for edges that require re-calculation, determining whether the edge is valid for contraction, before calculating the score in the same method as during initialization (except for boundary handling, as described below) and updating $U(e)$ to the current update cycle. In the case of edges that are no longer valid for contraction, or whose endpoints now map to the same vertex, we remove them from consideration altogether, by swapping such an entry with the entry at the bottom of the heap ($|H|-1$) and decrementing $|H|$, thus in effect removing it from the heap. We also consider the possibility of two different edges with endpoints mapping to the same vertices, for example, two edges of a triangle that has become degenerate from contraction of the other edge. We search for and store the pairs of mapped vertices in a binary tree, removing edges that map to a vertex pair that has already been stored in the tree.

For score re-calculation, we consider not only the immediate boundary change as in the initial calculation, but also the area that has been affected by previous contractions. We achieve this by storing this information with the retained edge during the contraction process in a binary tree. When we perform a contraction affecting the boundary, we obtain the vertex that becomes adjacent to the remaining vertex and the area immediately affected by the contraction, and then search the binary tree to obtain the affected areas previously associated with the affected edges (if any), and sum them to obtained the total area that has been affected overall, before using it to penalize the contraction and storing it with the retained edge, for use in later updates.

Using Figure 3-11 as an example, after v_m and v_n have previously been contracted, $<v_z, v_x>$ has the area of $<v_z, v_x, v_m>$ associated with it, while $<v_n, v_x>$ has

<v_x, v_y, v_n>'s area. Contracting <v_x, v_y> to v_y would then result in the sum of these two areas and <v_x, v_y, v_z> being associated with <v_z, v_y>. This method may in some cases overestimate the affected boundary area (for example, a boundary that alternates between convex and concave); however, this helps to further limit boundary change. Again, we skip the boundary check algorithm in models with few to no boundaries.

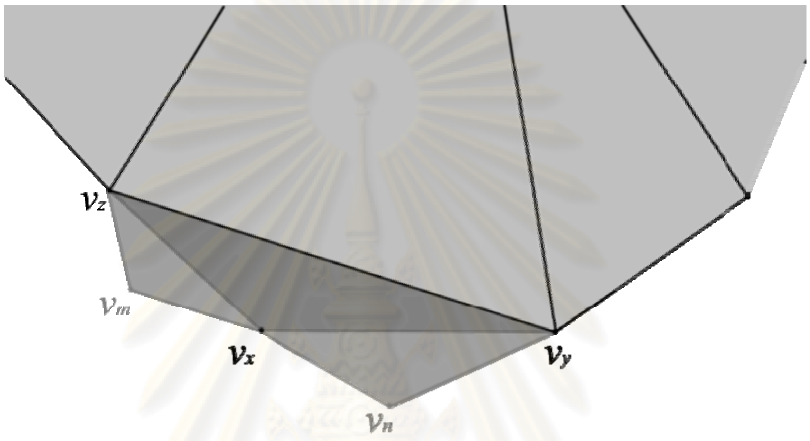


Figure 3-11: Boundary handling during score re-calculation

To save on unnecessary re-calculation when calculating the score, we use a novel caching method, by using a binary tree to store a given face's orientation and regularity after a given contraction. We key the information under the following convention: <f_x, v_x, v_y, U>. If the face is unchanged by a given contraction (e.g., f_x when contracting v_x to v_y, where f_x has v_y but not v_x), we store it (and search for it) under the convention <f_x, -1, -1, U>.

When we calculate a score and require the orientation and regularity of a given face after a given contraction, we determine whether the given face and contraction are already in the cache, and then whether update cycle stored in the entry is more recent than that of the face in question. If so, we retrieve the information and use it, before updating the most recent update cycle to the current value. Otherwise, we (re-)calculate the information, and either insert the information into the cache, keyed according to our convention (if it is not already in the cache), or update the existing cache entry (if it is already in the cache).

To limit the cache size, we have chosen to both perform an update on the whole heap and clear the cache when, at a given update, the size of the cache is 4 times the size of the number of the remaining faces, with a minimum of 10,000 cache entries at low numbers of vertices. (We also update the whole heap where there are less than 128 heap entries remaining.) This choice provides a good balance between memory usage and cache efficiency. We will also experiment with different ratios of cache entries to heap size. Figure 3-12 shows a summary of the process of updating each heap entry.

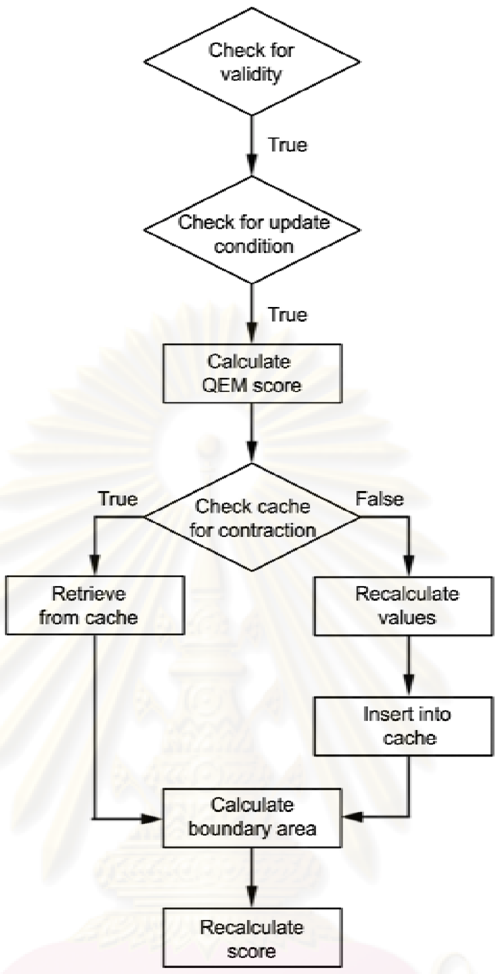


Figure 3-12: The heap updating process

After we have re-calculated entries in the heap as necessary, we restore the heap property, by swapping scores to their proper positions. We then repeat the cycle of edge contraction and heap updates, until the desired level of simplification has been reached, or in some cases, until no valid contractions remain in the heap.

3.6 Time and Complexity Analysis

Another point of analysis for our algorithm is the complexity of the algorithm, both time-wise and memory-wise. In a similar fashion to Garland and

Heckbert's analysis of their algorithm, it can be shown that, assuming an upper bound on the number of faces adjacent to any given vertex, edge-contraction based algorithms can generally be shown to have $O(n \log n)$ time complexity with respect to face count, as detailed below:

Initialization: Constructing the quadric matrices takes linear time ($O(n)$) with respect to face count, as well as calculating the normal vectors and the resulting curvatures of the vertices. Calculating the score for each entry takes constant time (assuming an upper bound on adjacent faces), and building the priority heap takes $O(n \log n)$ time.

Contraction: Assuming the upper bound on adjacent faces, a constant number of edges are affected with each contraction, and re-calculating the score for each edge takes constant time. It then takes $O(\log n)$ time (with respect to heap size) to restore the heap property per updated entry. Assuming that a constant number of faces k are removed between every update cycle, the total running time used for the simplification process, with respect to heap size, is:

$$\log n + \log (n-k) + \log (n-2k) + \ldots. + \log m = \log n! - \log m! = O(n \log n) \qquad (3\text{-}19)$$

Heap size is related to the number of edges. We have observed that in the models we have used, the number of vertices is approximately half of the face count. Assuming that the model is fully manifold, the number of edges is then approximately 1½ times the face count. Therefore, our algorithm takes $O(n \log n)$ running time in total.

Next, we will focus on memory complexity, with respect to the numbers of the various structural elements. For each face, we store the constituent vertices and edges along with its normal vector, nulling the information for each face as it becomes degenerate, and a Boolean variable storing whether it is degenerate. Assuming all-triangular meshes, this information is of constant size. Also, as mentioned earlier, we limit our cache size to a constant multiple of the number of remaining faces (or 10,000). This results in linear complexity storage ($O(n)$) with respect to original face count.

For each edge, we store its start and end vertices, its adjacent faces and a heap entry corresponding to the score of contracting the edge, with a constant size limit for each entry (assuming a fully-manifold model). Again, this results in linear complexity storage.

For each vertex, we store its coordinates, normal vector and curvature data, its adjacent faces, and its current quadric matrix (removing each vertex's data as it has been re-mapped via contractions). As mentioned in the Edge Contraction section, we also store bidirectional mapping information, reassigning and removing the data for re-mapped vertices. All of these data also have a constant storage size, and along with our bidirectional mapping scheme, this results in linear complexity storage. Assuming the number of edges is approximately 1½ times the face count as before, our algorithm uses linear storage with respect to face count. Both results are equal to that of Garland and Heckbert's QEM algorithm and most of its derivatives.

3.7 Summary

In this chapter, we have described our mesh simplification algorithm. We use a typical edge-contraction approach. We begin by calculating quadric matrices of each vertex, along with their principal curvatures and directions. After determining the validity of contracting each edge, we then use the quadric matrices to calculate the quadric error score, before determining the absolute normal curvature in the contraction direction and edge length, and multiplying these values with the quadric error. We then calculate regularity, along with angular and dihedral deviations based on each face's current and original orientation and each vertex's curvedness, and boundary changes where applicable, to calculate penalties to the error, before choosing to contract to the better scoring vertex and storing the scores in a priority heap.

When contracting edges, we keep track of vertex mappings and affected vertices, in order to assist in updating the heap. We use an updating scheme in which we only update the top portion of the heap when encountering a score that requires an update. We take overall boundary change into account when necessary. We also store a cache of orientations and regularities of faces resulting from possible contractions to assist in score re-calculation. We limit the size of the cache by clearing it when it reaches a certain size, and performing a full heap update. We contract edges until the desired level of simplification has been reached, or no more valid contractions remain.

We have analyzed the time and complexity for our algorithm. We have determined that initialization takes $O(n \log n)$ running time, and the simplification process also takes $O(n \log n)$ running time. We have also determined that the algorithm uses linear storage space in relation to face count, consistent with Garland and Heckbert’s original QEM algorithm and derivatives.

CHAPTER IV

EXPERIMENT AND RESULTS

The following chapter will describe our experiment with the new algorithm and the results obtained from the experiment. We will then discuss the results that we have obtained from the experiment.

4.1 Overview of the Experiment

In our experiment, we aim to compare the quality of the simplified models from our algorithm with that of Garland and Heckbert's quadric error metric method, by using both a visual and geometric comparison (using RMS of luminance difference and Hausdorff distances respectively). We also aim to verify how well the running times for our algorithm conform to the expected $O(n \log n)$ complexity that we have determined in Chapter III, and how much reduction in running time our partial updating scheme achieves. We also aim to determine the effects of our arbitrary choices for the heap-updating scheme.

4.2 Method

We have implemented the algorithm described in Chapter III using Microsoft Visual Basic .NET, and tested it on a Pentium Dual Core system with 2 GB RAM, using a data set of 388 models. 380 of the models used as sample data have been obtained from Princeton University's Benchmark for 3D Mesh Segmentation (Chen et al., 2009), with 5 other large models from the Georgia Tech University's Large Geometric Models Archive (Turbine Blade [Figure A-24(b)], Dragon [Figure

A-23 (c)], Horse [Figure A-24(a)], Canyon [Figure A-23(a)], Angel [Figure 43(b)]), the bunny model from the Stanford 3D Scanning Repository with holes in the bottom (Figure A-22(b)), the large standard Armadillo model from Stanford (Figure A-22(a)), and a dinosaur model (Figure A-23(b)) included to test for boundary handling and scalability. Details of these models can be found in the Appendix.

For each model, we perform the simplification algorithm, obtaining results at 50%, 20%, 10%, 5%, 2% and 1% of its original face count. As most of the models in the test sample have even vertex distribution, we test its performance on models with uneven vertex distribution by using QEM to simplify selected models to 50%, before using our algorithm.

As an alternative to logarithm-based penalizing, we have also decided to experiment with an alternate linear combination addition-based scoring method (as not all models require boundary handling, the weight for boundary penalties *β* will not be part of the linear combination). As our penalties are all scale-invariant, we will multiply them the length of the model's bounding-box diagonal *B*:

$$s_a\left(\langle v_x, v_y\rangle, v_x\right) = \varphi s'\left(\langle v_x, v_y\rangle, v_x\right) + \alpha\left(|B|\left(p_{ang} + p_{avg} + p_{rdc}\right)\right) + \delta\left(|B| p_{reg}\right) + \beta\left(|B| p_{bdr}\right) \tag{4-1}$$

$$\alpha + \delta + \varphi = 1 \tag{4-2}$$

For the purposes of running time analysis, we take the times from the fastest of three rounds of execution into consideration. We also perform the algorithm on selected models using a full heap update when encountering any score requiring an update, to determine the time savings from using our partial heap update scheme.

For comparison purposes, we use Garland and Heckbert’s QSlim program (a readily-available implementation of their algorithm) to simplify the same models to the same percentages using QEM, with a subset selection policy. For the models used to test uneven vertex distribution performance, we simplify to 50%, and then simplify the reduced model to the same percentages. After we have obtained the results of both mesh simplification methods, we render the results using VRMLView, and compare the results by using Cignoni, Rocchini and Scopigno’s (1998) Metro comparison program to obtain the *Hausdorff distance* between the original model and the simplified models. The Hausdorff distance between two models can be defined as:

$$d_H(X,Y) = \max\{\sup_{x\in X}\inf_{y\in Y} d(x,y), \sup_{y\in Y}\inf_{x\in X} d(x,y)\} \tag{4-4}$$

Where X and Y are the models to be compared, and $d(x, y)$ is the distance between two points x and y on the surfaces of X and Y respectively. In other words, the Hausdorff distance between two models is the largest distance between any two closest points on each model’s surface. This metric has been commonly used to assess the quality of simplified meshes. On the meshes reduced to 50% for uneven vertex distribution testing, we compare with the 50% reduced mesh, rather than the full version.

For visual comparison, we use the root-mean-square of luminance differences, a metric previously used by Lindstrom and Turk for their Image-Driven Simplification method. For simplicity, we will render the resulting models from a single representative angle, and then determine the root-mean-square of the difference between the luminance Y of each corresponding pixel:

$$RMS = \sqrt{\frac{\sum_{x}\sum_{xy}(Y_0(x,y) - Y_1(x,y))^2}{xy}} \tag{4-5}$$

It should be noted that while the Hausdorff distance is a definite indicator that applies to each model and can directly be compared, the RMS of luminance difference between a model and its reduced form will also depend on the rendering angle. Therefore, one should not compare the RMS results of different models; however, all renders of the same model from the same angle can be directly compared.

To determine the effects of the arbitrary values we have chosen in the heap-updating scheme on running time, we will replace the arbitrary values and run the algorithm on selected models to show the effects. We will run the algorithm by updating *n*-8 and *n*-4 layers, and clearing the cache and performing a full update when the cache is 6 and 2 times the size of the heap. We will also run the algorithm without using the caching method.

4.3 Experimental Results

The averaged graphical results of the Hausdorff distances from our algorithm and QEM are shown in Figure 4-1. Figures 4-2 to 4-4 shows Hausdorff results from the best and worst results from our data. Figures 4-5 and 4-6 show the running times for each model and LOD plotted against face count using only the Princeton data, as the non-Princeton data include models with about one order of magnitude more faces than the Princeton data, and the correlated trendlines, while

Figure 4-7 shows the plot when including non-Princeton data. Details of all the models that we have used along with the numerical results from Metro for each model, the RMS averages of the luminance differences of each model, and selected visual results from VRMLView and selected graphical Hausdorff results, are shown in Appendix A.

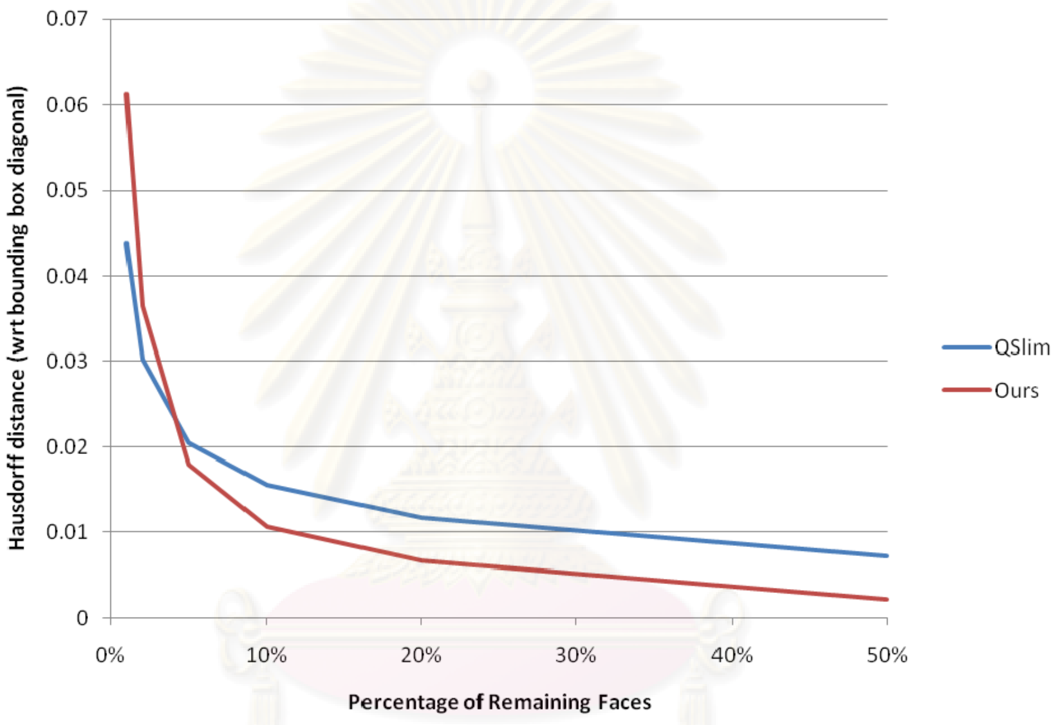


Figure 4-1: Graph comparing average Hausdorff distances between QSlim and our method

Figure 4-1 plots the average Hausdorff distance with respect to bounding box diagonal from all of our 388 sample models against the level of simplification, between our method (red line) and QEM (blue line), with the horizontal axis representing the remaining percentages of the original face count at which we obtain our result data (1%, 2%, 5%, 10%, 20%, 50%), and the vertical axis representing the Hausdorff distance with respect to bounding box diagonal. We observe that on most of the sample data models, the Hausdorff distance monotonically decreases as the percentage of remaining faces increases, and vice versa. We also observe that the new algorithm produces lower average Hausdorff distances than with QSlim up to between 2% and 5% remaining faces. Lastly, we notice an increased acceleration in the Hausdorff distance at less than 10% remaining faces using our method.

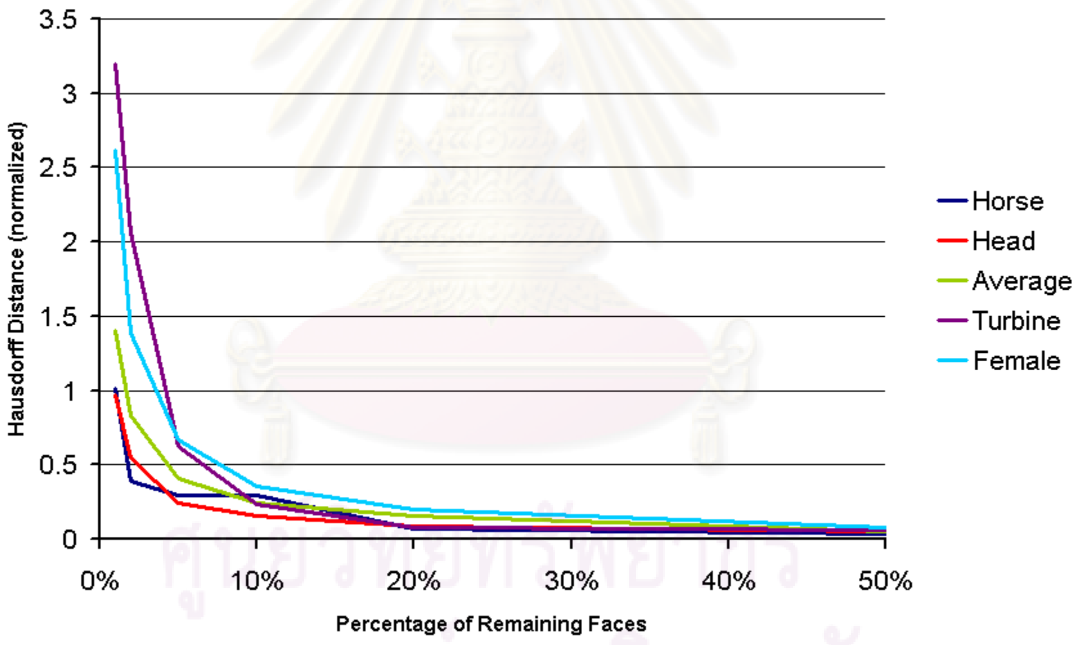


Figure 4-2: Graph comparing best and worst Hausdorff results with average for all results (normalized using 1% QEM distance)

Figure 4-2 shows a graph comparing the Hausdorff distance results of the models that produce the best and worst results compared to QEM, with the horizontal axis corresponding to remaining face percentages (as in Figure 4-1), and the vertical axis corresponding to the Hausdorff distance results, normalized by dividing with the Hausdorff distance of each given model's 1% QEM simplification. The best results (producing low Hausdorff distances and best visual resemblance) are with the horse (Figure A-22(a), blue line on graph) and one of the head models (#313 in the Princeton data, Figure A-21(a), red line on graph), while the worst results are with the turbine (Figure A-24(b), purple line on graph) and one of the low-polygon female models (#20, Figure A-14(b), cyan line on graph). We have also included the average Hausdorff distance, divided by the average 1% QEM distance (green line on graph).

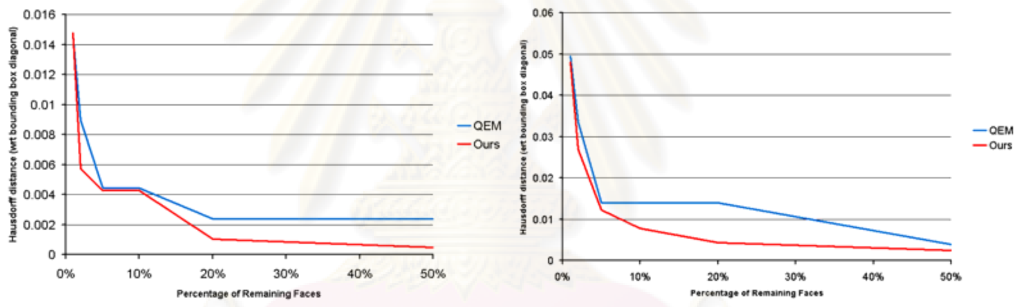


Figure 4-3: Hausdorff distances for best results: Horse (left) and Head #313 (right)

Figure 4-3 shows a comparison between the Hausdorff distances for the aforementioned best results (from Figure 4-2), the horse and head models between our method (red line) and QEM (blue line). We observe that the Hausdorff distance on these models using our method is better than, or at least comparable with, the distance using the QEM method, at all percentages of remaining face count.

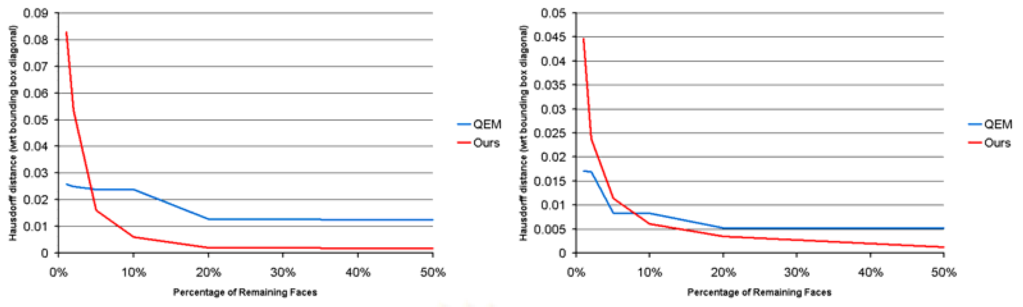


Figure 4-4: Hausdorff distances for worst results: Turbine Blade (left) and Female #20 (right)

Figure 4-4 shows a comparison between the Hausdorff distances for the aforementioned worst results (from Figure 4-2), the turbine blade and female models, between our method (red line) and QEM (blue line). We observe that, while the Hausdorff distance on the female model using our method is better than the distance using the QEM method above 10% remaining face count, it increases rapidly to become significantly higher than the corresponding distances for QEM from 5% downwards (similar to Figure 4-1). The turbine model produces better results down to 5% remaining face count, however, the Hausdorff distance from our method also increases rapidly from 5% downwards.

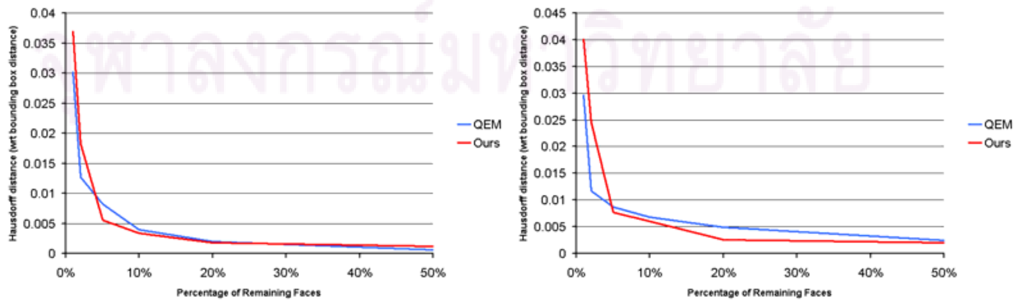


Figure 4-5: Hausdorff distances for average results: Teddy Bear #177 (left) and Princeton Armadillo #282 (right)

Figure 4-5 shows a comparison between the Hausdorff distances for two average results, for teddy bear model #177 and Princeton armadillo model #282, between our method (red line) and QEM (blue line). We observe that it both cases, our algorithm produces better or comparative results with QEM up to the 5%, however, the Hausdorff distance from our method also increases rapidly from 5% downwards.

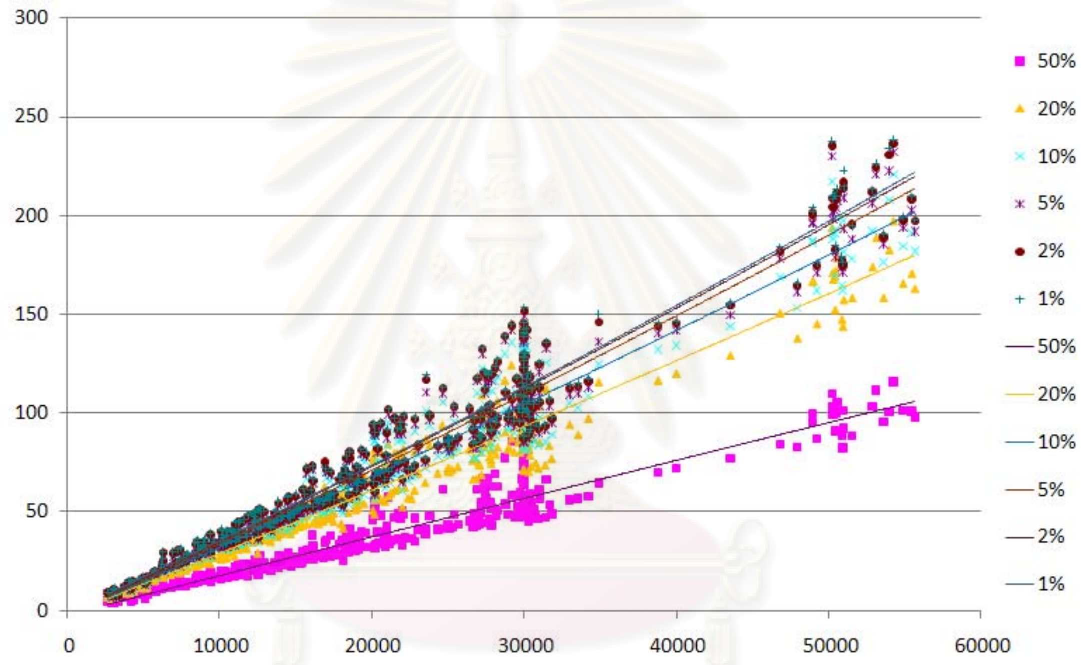


Figure 4-6: Runtimes for Princeton data plotted against face count with *n* log *n* trendlines

Figure 4-6 shows the runtimes for the Princeton data at the result percentages: 50% (magenta), 20% (orange), 10% (cyan), 5% (purple), 2% (red), 1% (blue) plotted against the original face count, with horizontal axis indicates the number of the original faces, and vertical showing the run time in seconds. We have also plotted trendlines in the same color as the data, and have shown the trendlines only in Figure 4-7. We observe that most of the data lies close to the respective trendlines, and the trendlines for the lower percentages of remaining faces lie closer to each other, as there are fewer faces to be reduced at those lower levels.

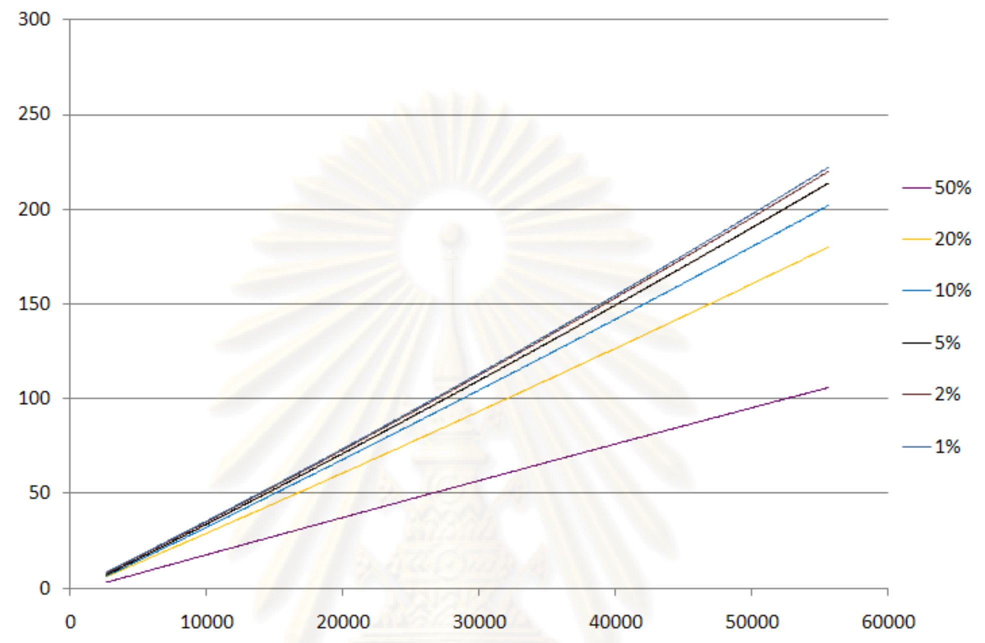


Figure 4-7: *n* log *n* trendlines for Princeton data on graph

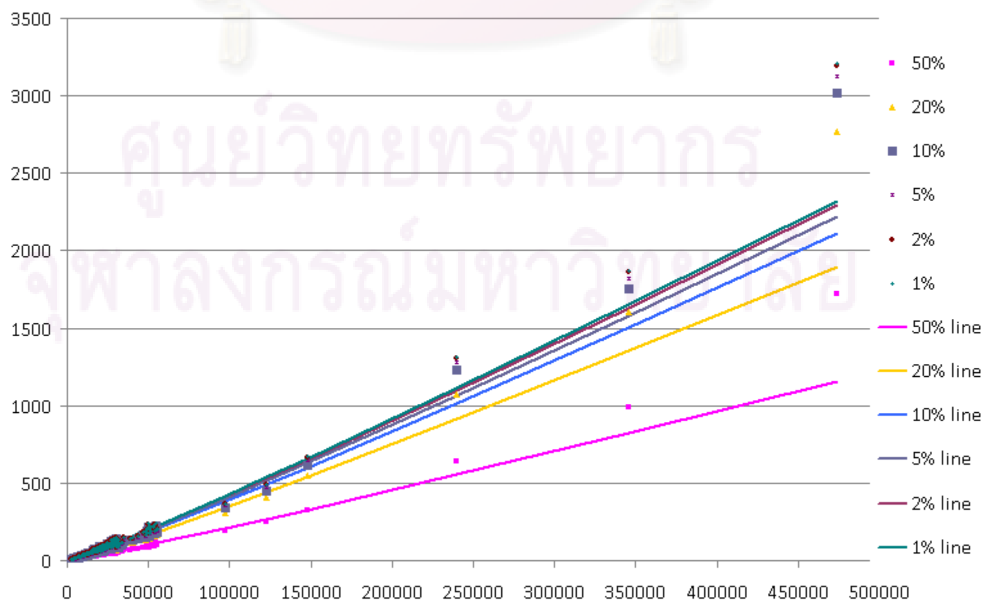


Figure 4-8: Runtimes for all data plotted against face count with *n* log *n* trendlines

Figure 4-8 shows the runtimes for all 388 models (including the larger non-Princeton models) at the result percentages plotted against the original face count as in Figures 4-6 and 4-7. We observe that, while our data generally lies relatively close to the trendlines, the largest models in our sample data (>240,000 faces: turbine, armadillo, and angel) produce significantly higher run times than the trendlines plotted from the $O(n \log n)$ time complexity determined in Chapter III.

4.4 Discussion

In this section, we discuss the running times and results as described in the previous sections, and determine the strengths and weaknesses of the algorithm. We begin by commenting on the Hausdorff and visual results, comparing the observed empirical running times with our complexity analysis and speculating on possible causes for outlying running times, and lastly, comparing the running times with and without the partial heap updating scheme.

4.4.1 Hausdorff and visual results

From the Hausdorff results as plotted in Figure 4-1, and visual results shown in the Appendix, we observe that our mesh simplification method shows a better or comparable performance to QEM at lower levels of simplification on average. Also, the RMS luminance difference from our algorithm is close to that from the QEM algorithm, suggesting that the factors we have implemented are useful for simplification at those stages. These results show that a combination of using the curvature measurement and angular deviations has improved the simplification results when simplifying to 5% of original face count. However, these factors seem to become less useful at more drastic levels of simplification, resulting in much higher Hausdorff distances than on models simplified using QEM. This shows that, although the curvature estimation is useful during the early stages of simplification, at later stages it may no longer be sufficiently accurate due to changes in the vertex’s locality. As a result, a contraction direction with low curvature, while sensible initially, may become less sensible at higher levels, as the contraction may involve features that were not accounted for in the original curvature calculation.

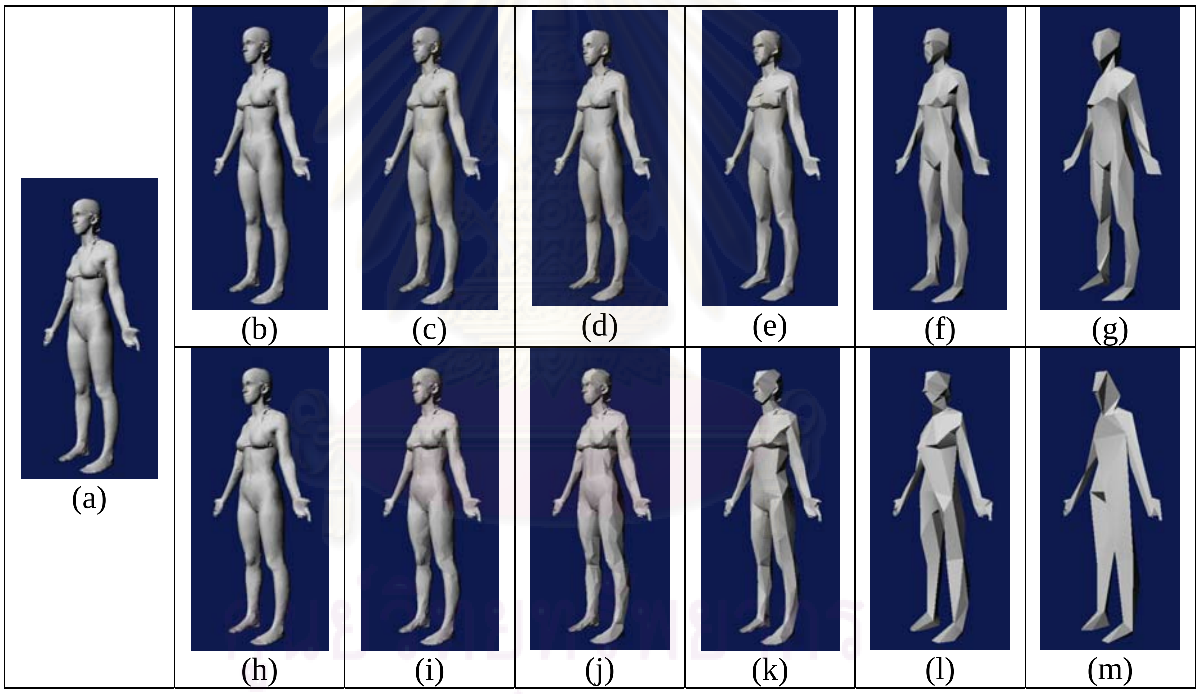


Figure 4-9: Comparison of female model (a) reduced with both QEM (b-g) and our methods (h-m)

As an example, in the female model from Princeton shown in Figure 4-9 at the 5%, 2%, and 1% levels (QEM: Figure 4-9(e)-(g), Ours: Figure 4-9(k)-(m)), a vertex in the knee area may become adjacent to a vertex at the ankle or waist areas, while the curvature measure was based on vertices in the knee area at the full face count level (shown in Figure 4-9(a)), without taking further areas into account. Therefore, a vertex in the knee area may be contracted to either the waist or ankle, based solely on the curvature of the knee area, resulting in higher Hausdorff distances than QEM. Comparing with QEM-based approaches, those that use the original model to penalize the quadric matrices (Kho and Garland, 2003, Jong et al., 2006, Li and Zhu, 2008) , or use larger matrices to take other factors into account (Wei and Lou, 2010) may use more triangles on feature areas (such as facial features and pointed fingers) than the smoother parts of the figure, like our algorithm, while those that use the current state of the mesh to calculate a penalty (Xu et al., 2008, Hussain, 2009, Tang et al., 2010), while also likely to preserve feature areas, may calculate a penalty based on a bad state.

For non-QEM-based approaches the female model, appearance-based methods (Cohen, 1996, Cohen et al., 1998 Lindstrom and Turk, 2000) are likely to generally produce better visual and Hausdorff distance results than QEM and our method, while memory-saving approaches (Lindstrom and Turk, 1998, Hussain et al., 2001) should perform about as well as QEM (and ours at lower levels). Balmelli et al.'s algorithm (2002) is designed only for 4-8 subdivided meshes where the subdivision hierarchy is known, and is thus irrelevant to our more general figures. Tang et al.'s global moment-based approaches (2007) are likely to produce similar Hausdorff distance results with QEM. Choi's optimal positioning approach (2008) produces more areas of high Hausdorff distance than with QEM, and is likely to produce worse results at lower level than with our algorithm.

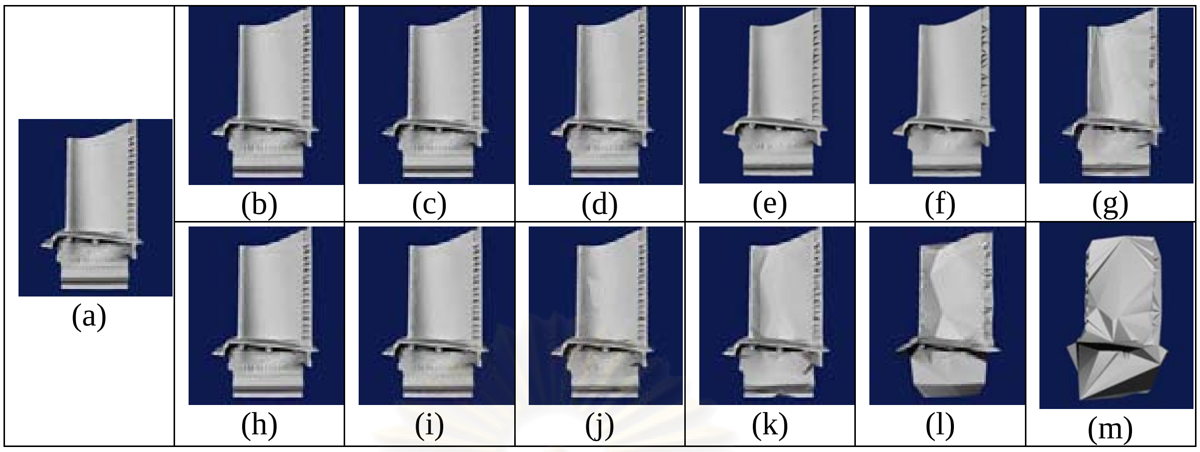


Figure 4-10: Comparison of turbine blade model (a), reduced with both QEM (b-g) and our methods (h-m)

Another of our worst cases is the turbine blade model (Figure 4-10). It has a complex structure, with inscribed lettering on the lower part of the model, and many faces hidden from view at all angles. We observe that using our method, the general shape of the blade has become highly corrupted at the 1% level (Figure 4-10(m)), and we also observe that the lettering on the lower part of the model gradually disappears at higher face count than with QEM. We believe that the algorithm may have used more faces on the non-visible portions of the model, resulting in less faces available for the visible parts of the model. QEM-based methods (Kho and Garland, 2003, Jong et al., 2006, Li and Zhu, 2008, Xu et al., 2008, Hussain, 2009, Tang et al., 2010, Wei and Lou, 2010), in concept, treat hidden faces equally as visible, and may end up retaining more faces on hidden surfaces than visible; although Kho and Garland's user-guided approach allows for the user to put a weight on the visible surfaces. Nevertheless, it is not likely for these algorithms to corrupt the blade model at low face counts.

In non-QEM based methods, most methods are also likely not to significantly corrupt the blade model at low face counts. Lindstrom and Turk's direct rendering approach, in particular, should work very well on models such as these, since contractions involving only hidden faces are likely considered not to have any cost as they do not affect the overall rendered image. It should be noted that Lindstrom and Turk use their image-driven method on this model, rendering both normally, and with the frontal faces culled.

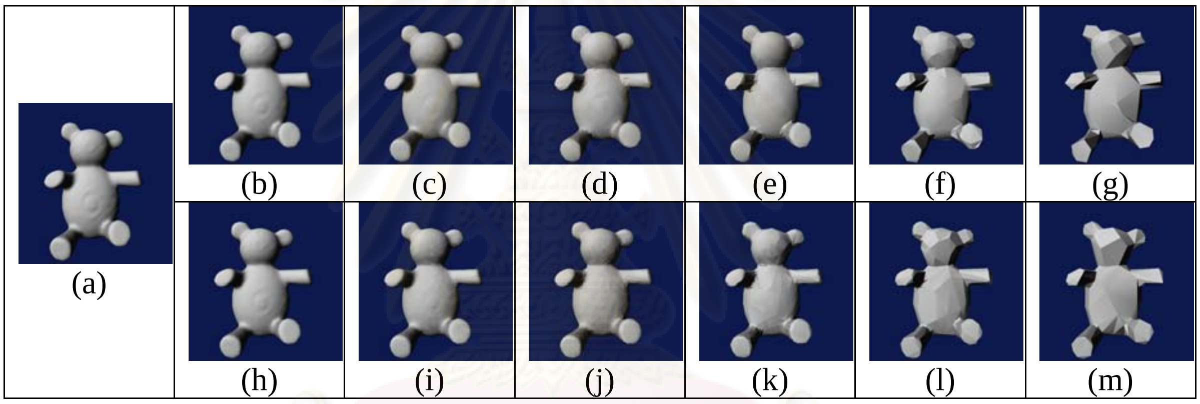


Figure 4-11: Comparison of teddy bear model (a), reduced with both QEM (b-g) and our methods (h-m)

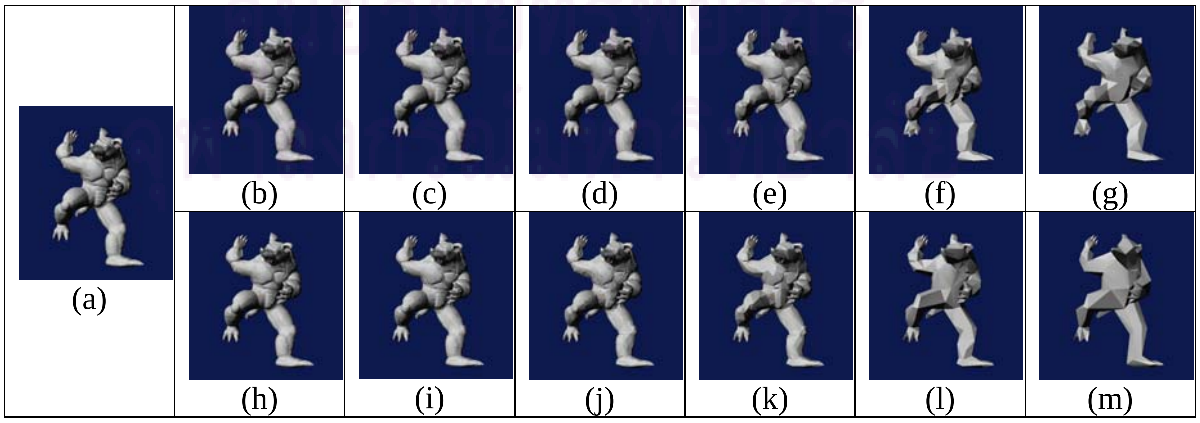


Figure 4-12: Comparison of Princeton armadillo model (a), reduced with both QEM (b-g) and our methods (h-m)

Two of our average cases are shown in Figures 4-11 and 4-12: a teddy bear model (#177) and one of Princeton's armadillo models (#282). In both models, the Hausdorff distance for our method starts out comparably with the results from QEM, up until 5% of original face count, where the Hausdorff distance from QEM becomes lower than that from our algorithm. The extensions used in QEM-based methods mostly focus on retaining features, while maintaining or improving on QEM's Hausdorff results, as the smooth surfaces can be represented with fewer faces without much surface error being introduced. Non-QEM-based methods are likely to produce similar Hausdorff distances to QEM, while the appearance-based algorithms, due to their focus on the overall appearance, should produce the best visual results.

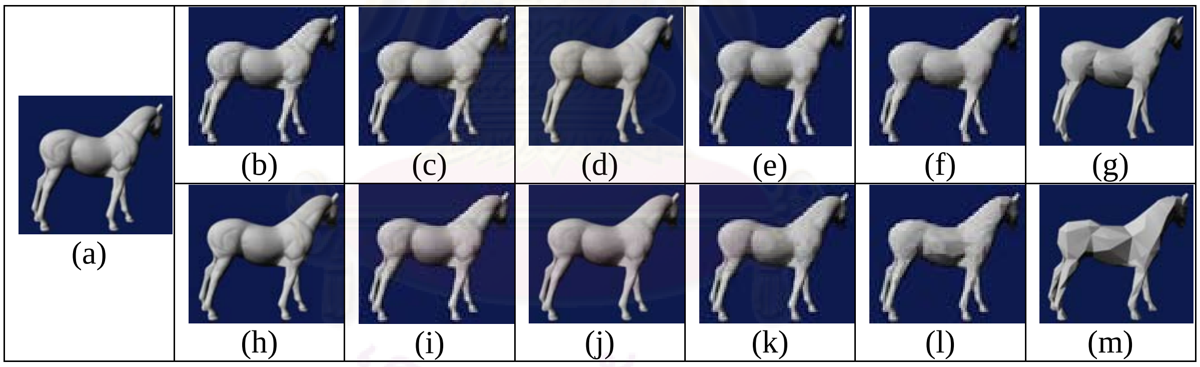


Figure 4-13: Comparison of horse model (a), reduced with both QEM (b-g) and our methods (h-m)

The best cases from our experiment are the horse model (Figure 4-13) and the head (#313) model (Figure 4-14). For the horse model, we observe that the model's surface is generally smooth all around, except for the ears. We also observe that our method preserves the ear's shape better at the 1% level of total faces, although it results in fewer faces being used for the rest of the horse's body (Figure 4-13(g) and (m)), resulting in a somewhat more faceted look than with QEM. QEM-based methods that focus on improving feature retention are likely to also devote more faces to smaller features in a similar fashion, while retaining a comparable Hausdorff distance performance to QEM. Most non-QEM-based methods are likely to produce a performance generally close to QEM, with appearance-based algorithms producing the best visual results, due to their focus on overall appearance.

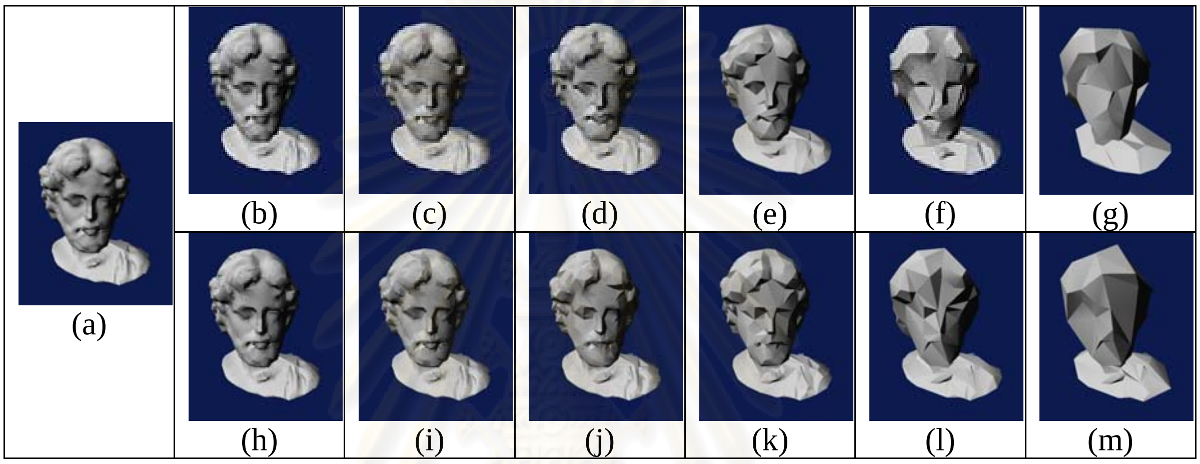


Figure 4-14: Comparison of head model (a), reduced with both QEM (b-g) and our methods (h-m)

For the head model (Figure 4-14), we observe that most of the model consists of relatively smooth surfaces, with some facial features. We observe that using both algorithms, the facial features have mostly disappeared between 2% and 1% remaining faces. We also notice that a small bump feature towards the bottom of the model is retained in our method all the way to 1% (Figure 4-14(m)), while in the QEM version, it has completely disappeared at the 1% level (Figure 4-14(g)). QEM-based methods that aim to retain features are likely to put a high score on various features in the model, including the facial features. As with the horse model, non-QEM-based methods should produce a performance of similar quality to QEM, with appearance-based algorithms producing the best visual results.

From the 50% reduced models, we observe that although the average Hausdorff distance is better when using our algorithm at all levels, after we have removed the outliers from consideration, the average of the remaining Hausdorff distances from our method is only comparable with QEM up to 20%, suggesting that uneven vertex distribution has a detrimental effect on the results of simplification.

When using the alternate linear combination addition-based penalizing method, we observe that on most of the models we have tested it with, the original logarithm-based penalizing method produces lower Hausdorff distances than when using the linear combination-based method. This suggests that, at least for our scale-invariant penalty factors, our logarithm-based penalizing method is more suitable than the linear combination addition-based method.

From the results we have observed, we conclude that our use of principal curvatures provides a useful indicator of the initial local properties of the surface at a given vertex point, and can help improve simplification performance at lower levels, and/or on models with smooth surfaces. Using both the orientation angle of each individual face and the dihedral angle between each pair of faces to calculate the overall error score of each given edge contraction has also helped faces to retain their orientation. Also, preventing the algorithm from contracting boundary vertices to non-boundary vertices, along with considering the change of area along the boundary, allows for the preservation of boundaries on models, such as in the canyon terrain model. Without this policy, boundaries would be much more noticeably eroded, resulting in higher Hausdorff distances. In the Appendix, Figure A-25 shows a side-by-side comparison of the canyon terrain model simplified to 1% with and without any form of boundary preservation, showing significant boundary erosion without boundary preservation, while Figure A-26 shows a comparison showing the holes on the bottom of the bunny model, simplified to 1% with and without boundary preservation, with the holes being better preserved when using boundary preservation.

Another possible weak point is the use of a curvedness-inverse-weighted average of the angle between the normal vectors of the resulting face and its vertices as an indicator of orientation quality, as it may only provide a partial indication of how well a face's orientation fits the surface with the given vertices. Although this method works well on smooth surfaces, it does not consider "noise" (such as minor projections or corners) in between the vertices (the canyon terrain model provides an example). One possible explanation is that some contractions may result in triangles with a surface covering the area with the noise, and its vertices on the smoother surrounding surface, resulting in an average suggesting that the resulting triangle has a near-ideal orientation, thus producing a low penalty. Another possibility is all three vertices lying on areas of high curvedness (for example, facet edges of a box), producing similar results.

4.4.2 Comparing empirical running time to complexity analysis

To determine how well the time spent on simplification for model conforms to the expected $O(n \log n)$ running time complexity, we have divided the running time at each level of detail by $n \log n$ (where n is the face count), along with plotting the results on a graph and determining trendlines, as shown in Figures 4-5 to 4-7. From the graphs, we observe that the simplification of most models conforms well to the expected complexity, with a few outliers. Also, the trendlines for later stages of simplification are closer, as there are fewer faces between each stage. The maxima, minima, arithmetic means and standard deviations of the results for *t*/*n* log *n* at each level of detail are shown in Table 4-1.

Table 4-1: Maxima, minima, arithmetic means and standard deviations for *t*/*n* log *n* at each LOD

LOD	**Max**	**Min**	**Mean**	**Σ**
50%	0.000701	0.000310	0.000430	0.000062
20%	0.001100	0.000518	0.000705	0.000089
10%	0.001202	0.000599	0.000786	0.000097
5%	0.001252	0.000632	0.000826	0.000100
2%	0.001282	0.000647	0.000852	0.000103
1%	0.001288	0.000654	0.000862	0.000103

Comparing the actual runtimes with the means and standard deviations, we have noticed that the following models produce running times significantly higher ($\frac{(t/n\log n)-\bar{x}}{\sigma}$>+1) than average: 1 11-14 47 100 101 103-121 124 126 128 131 134 136 141 143 144 152 156 158 160 161 191 211 231 241-244 246 247 249-251 266 301-303 308-311 314 315, Angel, Armadillo, Turbine blade. The following models produce significantly lower running time ($\frac{(t/n\log n)-\bar{x}}{\sigma}$<−1): 2 5 6 8 16-21 35-38 40 154 183 270 275 276 284 288 307 322 323 328 329 333 335 336 343-345 348-350 354 357 363 365-367 369 374 375.

Possible explanations for the discrepancy in running time for the aforementioned figures include:

Changing system workloads during the simplification of each model: Although we have tried to minimize the effects of different workloads affecting the execution of the algorithm by taking the best running times from multiple executions of the algorithm on each model, fluctuations in the system workload and memory usage during the process may still cause some change in overall execution speed, especially in cases involving very large models and long execution time. The three largest models in our test data (all from non-Princeton data) have significantly longer running time, especially so with the angel model (the largest of our data).

Frequency of updates: The metric and (to some extent) the updating procedure control the order in which edges are selected for contraction, and on some models, it may result in more frequent updates than average, resulting in longer execution time (due to the overhead of searching through the heap during updates). This may depend on the overall facial structure of each model. Although there are no obvious patterns on how a model may produce such a result, we have observed that many of the chair, box and bird models produce significantly higher running time, suggesting the facial structure of those models result in contraction sequences that produce frequent updates; while the four-legged animal and trophy models produce lower running time (significantly in many cases), possibly due to lengthy sequences of contractions between updates.

We expect that all QEM-based approaches should produce similar running times to our approach, due to the calculated time complexity. Among non-QEM-based approaches, the appearance-preserving algorithms either pre-process the mesh to create simplification envelopes or parameterized maps, or use a rendering-based approach to calculate the score, thus using more time than QEM-based algorithms. Other non-QEM-based algorithms still use geometrically based factors to calculate the score, and thus should also produce similar running times.

4.4.3 Analysis of running times with full and partial heap updates

Comparing the running times between simplifying with full and partial heap updates at every score to be updated, we observe a speed up by a factor of 5.38 on average. The speed up on the overall running time on our selected models ranges from 4.253 (#391) to 7.734 (#141). The models we have selected and direct comparisons of the running time are shown in Table A-2 (in the Appendix). These results show that the partial heap updating scheme significantly reduces running time. The reason behind the running time reduction is that our scheme generally only inspects the topmost portion of the heap for updates, with contractions at the bottom of the heap not being checked (except during the occasional full heap update), and edges that would normally be updated many times before it is encountered in the heap may only need to be updated a few times before contraction.

4.4.4 Analysis of running times using different update parameters

Comparing the running times when using different values than we use normally (see Table A-7), we observe the following: increasing the size of the cache before a full update reduces running time, and vice versa, due to having fewer full updates during the execution. However, we believe that the increased cache size may also result in memory problems when reducing larger models than those used to test for this section. Also, performing the algorithm without the cache tends to use more running time than with the cache.

Updating more layers significantly increases running time, due to the extra overhead of updating more heap entries; however, updating fewer layers mostly produces running times similar to using the normal values. We also note, however, that one model (#111) does not reduce to 2% when using fewer layers, whereas it does when using the normal values.

4.5 Summary

In this chapter, we have performed an experiment to compare the results of our algorithm with Garland and Heckbert’s QEM method, both time-wise and performance-wise (based on the Hausdorff distance between the original and simplified models). We have used a sample array of 388 models to test the algorithm, with 380 models from a sample set by Princeton, with other models from Georgia and Stanford included for scalability and boundary handling testing. Each model was reduced to 50%, 20%, 10%, 5%, 2% and 1%, rendered, and compared to the original to determine Hausdorff distance and RMS average of luminance difference. We also reduced some selected models to 50% using QEM, before then using both of our algorithms, to test its relative performance on meshes with uneven vertex distribution. The QSlim implementation was used to provide the results from the QEM method. According to the experimental results, the average Hausdorff distance using our algorithm on the full-sized models is lower than QEM down to between 5% and 2% of the original face count, with RMS average results similar to those from WEM; however, on meshes with uneven vertex distribution, the average distance when using

QEM is significantly better than our algorithm from 10% downwards. We also observe that on most models, the Hausdorff distance increases faster after reducing to 10% remaining faces. We observe that our best results occurring from models with mostly smooth surfaces, with the worst results occur with a model with significantly sharp features and a model with hidden surfaces, with most other models producing average results. The best models produce results better than or comparable to QEM at all percentages of face count, while the worst models produce Hausdorff results many times higher than QEM at 1% face count. We also observe that the runtimes generally conform to the $O(n \log n)$ time complexity. However, we note that the largest data we have used has significantly higher running time than the trendlines suggest.

The results from our algorithm are worse than QEM at higher rates of simplification, as the curvature measurement may no longer be accurate at more drastic levels of simplification. Comparing the results to other methods we have referenced, QEM-based methods that focus on the retention of features may use more triangles on areas deemed more visually important than with QEM. Among non-QEM methods, our proposed method should produce better visual resemblance than global-moment based approaches, optimal placement, and memory-saving approaches, although appearance-based algorithms should still produce better visual resemblance, especially in cases where hidden surfaces are involved (such as the turbine blade in Figure 24). Another possible weak point may be that the curvedness-inverse-weighted average of vertex normals may provide only a partial indication of orientation quality.

Although most of our runtimes conform to $O(n \log n)$ time complexity, we observe that many of the models have significantly different running times from the expected result. We have theorized two possible causes, changes in system loads during the execution, and frequency of heap updates (which may result from facial structures). Runtimes should be comparative to most QEM-based methods, and non-appearance-based non-QEM-based methods, while using less time than the appearance-based methods. We observe that using the original logarithm-based penalizing method produces lower Hausdorff distances than using the linear combination addition-based penalty.

We also observe that our heap updating scheme produces a significant reduction in running time against full updates. We also tested the algorithm when changing our chosen values for number of updated layers and cache size, and note that increasing the number of layers also increases the running time, although reducing the number of layers produces similar running time. Using the cache usually reduces running time, and increasing the cache size before clearing the cache and performing a full heap update significantly reduce the running time, and vice versa; nevertheless, there may be memory issues when reducing large models.

CHAPTER V

CONCLUSIONS

In this chapter, we will summarize the purpose of our research and the results we have obtained from it. We will then also consider the weaknesses of some of our results, and suggest possibilities for future research into reducing these weaknesses.

5.1 Summary

Mesh simplification is a procedure to simplify the facial complexity of three-dimensional polygonal meshes for easier general handling, while retaining as much resemblance to the original mesh as possible. As finding the optimal simplification has been shown to be an NP-Hard problem, much research has been made into determining heuristics that improve the various performance aspects of mesh simplification algorithms. Mesh simplification algorithms based on a vertex contraction mechanism, such as Garland and Heckbert’s well-known Quadric Error Metric method, have become popular for research, because such a mechanism lends naturally to level-of-detail based structures, which allow for a model to be displayed with a relatively exact number of faces, thereby allowing for better allocation of faces for rendering.

Many papers have described improvements to the Quadric Error Metric method to extend it from being based solely on geometric distance, while others use a non-QEM-based scoring method. Several papers improving the QEM method use the local geometry of the area to calculate a single curvature-like measure for each vertex to assist in the scoring. However, we believe that using a single measure may be insufficient in describing the locality around a given vertex, as the surrounding surface can be properly described using maximum and minimum curvatures ($k_{\max}$ and $k_{\min}$), along with principal directions. We also believe that contracting an edge in a direction of low curvature is less likely to cause significant visual changes than contracting in a direction of high curvature. Therefore, our paper presents a method to use these properties to create an improved mesh simplification method based on the edge contraction mechanism.

Our method aims to simplify the possible ambiguity of using a single measure of curvedness on a given vertex by incorporating the calculation of the principal curvatures and their directions to assist in determining the curvature in a given edge's direction, to produce low scores on contracting edges with low curvature and high scores on edges with high curvature, from the same given vertex. We also use the curvatures to help provide a measure of how well a resulting face would fit the surface, given its constituent vertices and their normal vectors, by calculating a curvedness-inverse-weighted average of the resulting face's normal vector with the vertices' normal vectors.

Besides the curvature-based measures, for each resulting face, we also take into account its regularity, its relative orientation to its original orientation, and its relative dihedral angle with adjacent faces, and use the worst case of each for the score calculation. We also include a simple boundary preservation policy, by disallowing the contraction of boundary vertices to non-boundary vertices, and considering the change in boundary area when calculating the error score. This measure has been shown to provide preservation for boundaries and holes.

In our implementation of this algorithm, we use some methods to reduce time used for heap updating, namely, updating only the top portion of the heap, and using a cache to store the normal vector and regularity of resulting faces to assist in the calculation of scores. We have shown that the overall algorithm takes $O(n \log n)$ running time and linear storage relative to face count.

Testing our algorithm on 388 models (mostly from Princeton data), we have found that this approach produces lower Hausdorff distances than the QEM method at lower levels of simplification and/or on models with smooth surfaces, suggesting that the use of a direction-based curvature measurement can provide some improvement to the simplification in the early stages. However, the QEM algorithm still produces lower distances at more drastic levels, especially on meshes with uneven vertex distribution, suggesting that the factors we have used to assist the simplification become deficient in later stages, likely due to the increase in the area covered by the faces adjacent to any given vertex. Also, using the curvedness-inverse-weighted average of angles between the face and its vertices’ normal vectors, while working well on smooth surfaces, may not take surface noise, such as minor projections, into account. We also observe that our choice of logarithm-based penalty generally produces better Hausdorff results than the linear combination addition-based penalty described in Chapter IV.

We have also found that the running time from our algorithm mostly conforms to the expected $O(n \log n)$ complexity, with a few outliers, although the models with the highest face count produce significantly higher running times. Comparing the running time results between using and not using our partial heap updating scheme, we have observed a significant decrease in running time. We note that increasing the number of layers updated at each update increases the running time; however, the running time remains similar when reducing the number of layers. We also note that our caching method reduces running time, and increasing the cache size reduces running time, and vice versa.

5.2 Future Work

Although our algorithm produces better Hausdorff distances than QEM in the early stages of simplification, QEM still produces better results at more drastic levels, especially on meshes with uneven vertex distribution. Therefore, our future work to improve the algorithm includes improving the curvature factors that we use for score calculation to make them more tolerant to uneven vertex distribution, surface noise and changes in the model during the later stages of simplification. Other possibilities are including easily-calculated factors that remain relatively unaffected by these issues in the algorithm, and improving the robustness of the curvedness-inverse-average of normal vectors as an indicator of ideal facial orientation.

Another possible topic of research is improving the memory management of the process. Our paper uses various arbitrary values that we consider provide a good balance for our purposes; however, further research into determining the best values for the best balance of performance may be required. Also, further research into determining the best method for applying the penalties to the QEM score may improve results.

REFERENCES

Agarwal, P.K., and Suri S. Surface approximation and geometric partitions. In **SODA '94: Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms**, 24-33. Philadelphia : SIAM, 1994.

Balmelli, L., Vetterli, M., and Liebling, T.M. Mesh optimization using global error with application to geometry simplification. **Graphical Models** 64, 3-4 (May-July 2002) : 230-257.

Batagelo, H.C., and Wu, S-T. Estimating curvatures and their derivatives on meshes of arbitrary topology from sampling directions. **The Visual Computer** 23, 9 (September 2007) : 803-812.

Baumgart, B.G. A Polyhedron Representation for Computer Vision. In **AFIPS '75: Proceedings of the May 19-22, 1975, national computer conference and exposition**, 589-596. New York : ACM, 1975.

Boubekeur, T., and Alexa, M. Mesh simplification by stochastic sampling and topological clustering. **Computers & Graphics** 33, 3 (2009) : 241-249.

Chen, H-H., Luo, X-N., and Ling, R-T. Surface Simplification Using multi-edge mesh collapse. In **ICIG '07: Proceedings of the Fourth International Conference on Image and Graphics**, 954-959. Washington : IEEE Computer Society, 2007.

Chen, H-K., Fahn, C-S., Tsai, J.J.P., Chen, R-M., and Lin, M-B. A linear time algorithm for high quality mesh simplification. In **IEEE Sixth International Symposium on Multimedia Software Engineering, 2004. Proceedings.** 169-176. Los Alamitos, CA : IEEE Computer Society, 2004.

Chen, X., Golovinskiy, A., and Funkhouser, T. A benchmark for 3D mesh segmentation. In ACM SIGGRAPH 2009 papers, 73:1-73:12. New York : ACM, 2009.

Choi, H.K., Kim, H.S., and Lee, K.H. A mesh simplification method using noble optimal positioning. In Falai Chen and Bert Jüttler (eds.), **GMP'08: Proceedings of the 5th international conference on Advances in geometric modeling and processing**, 512-518. Heidelberg : Springer-Verlag Berlin, 2008.

Cignoni, P., Montani, C., and Scopigno, R. A Comparison of Mesh Simplification Algorithms. **Computers & Graphics** 22, 1 (1998) : 37-54.

Cignoni, P., Rocchini, C., Montani, C., and Scopigno, R. External memory management and simplification of huge meshes. **IEEE Transactions on Visualization and Computer Graphics**, 9, 4 (October-December 2003) : 525-537.

Cignoni, P., Rocchini, C. and Scopigno, R. Metro: measuring error on simplified surfaces. **Computer Graphics Forum** 17, 2 (June 1998) : 167-174.

Clark, J.H. Hierarchical geometric models for visible surface algorithms. **Communications of the ACM** 19, 10 (October 1976) : 547-554.

Cohen, J., et al. Simplification envelopes. In **SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques**, 119-128. New York : ACM, 1996.

Cohen, J., Manocha, D., and Olano, M. Simplifying polygonal models using successive mappings. In Roni Yagel and Hans Hagen (eds.), **VIS '97: Proceedings of the 8th conference on Visualization '97**, 395-ff. Los Alamitos, CA : IEEE Computer Society Press, 1997.

Cohen, J., Olano, M., and Manocha, D. Appearance-preserving simplification. In **SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques**, 115 - 122. New York : ACM, 1998.

Eck, M., et al. Multiresolution Analysis of Arbitrary Meshes. In **SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques**, 173-182. New York : ACM, 1995.

Fahn, C-S., Chen, H-K., and Shiau, Y-H. Polygonal Mesh Simplification with Face Color and Boundary Edge Preservation Using Quadric Error Metric. In **MSE '02: Proceedings of the Fourth IEEE International Symposium on Multimedia Software Engineering**, 174. Washington : IEEE Computer Society, 2002.

Garland, M., and Heckbert, P.S. Simplifying surfaces with color and texture using quadric error metrics. In **VIS '98: Proceedings of the conference on Visualization '98**, 263 - 269. Los Alamitos, CA : IEEE Computer Society Press, 1998.

Garland, M., and Heckbert, P.S. Surface simplification using quadric error metrics. In **SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques**, 209 - 216. New York : ACM, 1997.

Gieng, T.S., Hamann, B., Joy, K.I., Schussman, G.L., and Trotts, I.J.. Smooth hierarchical surface triangulations. In Roni Yagel and Hans Hagen (eds.), **VIS '97: Proceedings of the 8th conference on Visualization '97**, 379 - 386. Los Alamitos, CA : IEEE Computer Society Press, 1997.

Guèziec, A. Surface simplification with variable tolerance. In **Second Annual Intl. Symp. on Medical Robotics and Computer Assisted Surgery (MRCAS '95)**, 132-139. 1995.

Hamann, B. A data reduction scheme for triangulated surfaces. **Computer Aided Geometric Design** 11, 2 (April 1994) : 197 - 214.

Hoppe, H. Progressive meshes. In **SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques**, 99-108. New York : ACM, 1996.

Hussain, M. Efficient Simplification Methods for Generating High Quality LODs of 3D Meshes. **Journal of Computer Science and Technology** 24, 3 (2009) : 604-inside back cover.

Hussain, M., Okada, Y., and Niijima, K. Fast, Simple and Memory Efficient Mesh Simplification. In **Proceedings of the Fourth International Conference on Computer Graphics and Imaging (CGIM2001)**, 72-77. Anaheim : IASTED/Acta Press, 2001.

Jia, S., Tang, X., and Pan, H. Fast Mesh Simplification Algorithm Based on Edge Collapse. **Lecture Notes in Control and Information Sciences** 344 (2006) : 275-286.

Jong, B-S., Tseng, J-L., and Yang, W-H. An efficient and low-error mesh simplification method based on torsion detection. **The Visual Computer** 22, 1 (January 2007) : 56-67.

Kho, Y., and Garland, M. User-guided simplification. In **I3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics**, 123 - 126. New York : ACM, 2003.

Kim, H.S., Choi, H.K., and Lee, K.H. Mesh simplification with vertex color. In Falai Chen and Bert Jüttler, **GMP'08: Proceedings of the 5th international conference on Advances in geometric modeling and processing**, 258-271. Heidelberg : Springer-Verlag Berlin, 2008.

Kobbelt, L., Campagna, S., and Seidel, H-P. A General Framework for Mesh Decimation. In Torsten Moller and Colin Ware (eds.), **Proceedings of Graphics Interface**, 43-50. Lethbridge, Alberta : AK Peters/CRC Press, 1998.

Kovalevsky, V.. Algorithms and data structures for computer topology. In Gilles Bertrand, Atsushi Imiya and Reinhard Klette (eds.), **Digital and image geometry**, 38-58. New York : Springer-Verlag New York, 2001.

Levoy, M., et al. The digital Michelangelo project. In **SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques**, 131-144. New York : ACM Press/Addison-Wesley, 2000.

Li, Y., and Zhu, Q. A New Mesh Simplification Algorithm Based on Quadric Error Metrics. In **ICACTE '08: Proceedings of the 2008 International Conference on Advanced Computer Theory and Engineering**, 528-532. Washington : IEEE Computer Society, 2008.

Lindstrom, P. Out-of-core simplification of large polygonal models. In **SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques**, 259-262. New York : ACM Press / Addison-Wesley, 2000.

Lindstrom, P., and Turk, G. Fast and memory efficient polygonal simplification. In **VIS '98: Proceedings of the conference on Visualization '98**, 279 - 286. Los Alamitos, CA : IEEE Computer Society Press, 1998.

Lindstrom, P., and Turk, G. Image-driven simplification. **ACM Transactions on Graphics (TOG)** 19, 3 (July 2000) : 204-241.

Luebke, D.P. A Developer's Survey of Polygonal Simplification Algorithms. **IEEE Computer Graphics & Applications** 21, 3 (May 2001) : 24-35.

Ripolles, O., Chover, M., Gumbau, J., Ramos, F. and Puig-Centelles, A. Rendering continuous level-of-detail meshes by Masking Strips. **Graphical Models** 71, 5 (September 2009) : 184-195.

Rossignac, J., 3D compression made simple: Edgebreaker with Zip&Wrap on a corner-table. In **International Conference on Shape Modeling and Applications, SMI 2001**, 278-283. Los Alamitos, CA : IEEE Computer Society Press, 2001.

Rossignac, J., and Borrel, P. Multi-resolution 3D approximations for rendering complex scenes. In B. Falcidieno and T.L. Kunii (eds.), **Geometric Modeling in Computer Graphics**, 455-465. Genova, Italy : Springer Verlag, 1993.

Schroeder, W.J., Zarge, J.A., and Lorensen, W.E. Decimation of triangle Meshes. In **SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques**, 65-70. New York : ACM, 1992.

Smith, C. **On vertex-vertex systems and their use in geometric and biological modelling**. Doctoral dissertation, University of Calgary, 2006.

Tang, H., Shu, H.Z., Dillenseger, J.L., Bao, X.D., and Luo, L.M. Technical Section: Moment-based metrics for mesh simplification. **Computers and Graphics** 31, 5 (October 2007) : 710-718.

Tang, Z., Yan, S., and Lan, C. A New Method of Mesh Simplification Algorithm Based on QEM. **Information Technology Journal** 9, 2 (2010) : 391-394.

Varakorn Ungvichian and Pizzanu Kanongchaiyos. Mapping A 3-D Model into Abstract Cellular Complex Format. **Computer-Aided Design and Applications Journal** 3, 1-4 (2006) : 395-404.

Vieira, A.W., Velho, L., Tavares, G., and Lewiner, T. Fast stellar mesh simplification. In **XVI Brazilian Symposium on Computer Graphics and Image Processing, 2003. SIBGRAPI 2003.**, 27-34. Los Alamitos, CA : IEEE Computer Society, 2003.

Wei, J., and Lou, Y. Feature Preserving Mesh Simplification Using Feature Sensitive Metric. **Journal of Computer Science and Technology** 25, 3 (2010) : 595-605.

Wu, J., and Kobbelt, L. Fast Mesh Decimation by Multiple-Choice Techniques. In Günther Greiner (ed.), **Vision, modeling, and visualization 2002**, 241-248. Berlin : Aka GmbH, 2002.

Xu, L., Chen W., Liu. J., and Lü, T. An improved quadric error metrics based on feature matrix. In **2008 IEEE Conference on Robotics, Automation and Mechatronics**, 582. Chengdu, China : IEEE, 2008.

Zelinka, S., and Garland, M. Permission grids: practical, error-bounded simplification. **ACM Transactions on Graphics (TOG)** 21, 2 (April 2002) : 207-229.

Zhigeng, P., Jiaoying, S., and Kun, Z. A new mesh simplification algorithm based on triangle collapses. **Journal of Computer Science and Technology** 16, 1: (January 2001) : 57-63.

APPENDIX

EXPERIMENTAL RESULTS

In this chapter, we will display the complete running time and comparison of Hausdorff distances between QEM and our algorithm. We will then display selected visual results, as well as other images related to our results.

A.1 Hausdorff distance and luminance difference results

In Table A-1, we display the Hausdorff distances, with respect to bounding box diagonal, of the models in our sample data simplified with QEM (top) and our algorithm (middle), along with the running time (bottom). Table A-2 will display the parameters used to weight the penalties.

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm

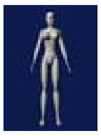
Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
1	9408	0.003054	0.011327	0.012432	0.017059	0.040900	0.040838
		0.002224	0.005295	0.012277	0.032251	0.043783	0.055846
		19.158	33.048	35.902	36.953	38.445	39.006
2	20096	0.001073	0.003161	0.004794	0.007247	0.013983	0.027762
		0.001112	0.003512	0.005974	0.010926	0.017590	0.031017
		32.236	50.753	58.965	62.109	64.162	65.214
3	11278	0.001092	0.006000	0.005410	0.009275	0.019970	0.032207
		0.001957	0.004880	0.010716	0.025016	0.067621	0.073772
		19.869	33.909	36.993	38.585	40.068	40.759

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

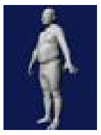
Model	Faces	50%	20%	10%	5%	2%	1%
4	11348	0.001332	0.003302	0.007753	0.012306	0.020742	0.042905
		0.001747	0.005100	0.010210	0.034277	0.053011	0.079049
		18.507	31.465	34.049	35.211	36.172	36.583
5	30308	0.003292	0.003839	0.007148	0.006391	0.027964	0.028467
		0.001560	0.002181	0.004314	0.010219	0.034044	0.059597
		45.405	70.301	81.397	87.045	89.068	89.859
6	20192	0.001643	0.003935	0.005358	0.009647	0.016934	0.036171
		0.001740	0.004245	0.007328	0.017097	0.061595	0.074193
		31.916	48.810	56.101	58.694	60.257	60.647
7	16878	0.006656	0.006656	0.013176	0.017469	0.035486	0.063384
		0.001441	0.002913	0.006624	0.020418	0.042857	0.062437
		27.920	46.206	50.202	52.986	54.789	55.410
8	22026	0.000985	0.004734	0.006311	0.009182	0.017323	0.025914
		0.001144	0.002262	0.004779	0.009860	0.026979	0.047495
		33.308	52.455	60.958	65.124	66.506	67.287
9	5274	0.007875	0.011246	0.022751	0.025450	0.064382	0.158390
		0.002874	0.010769	0.013104	0.033556	0.043925	0.099771
		8.262	11.927	13.650	14.411	15.352	15.773
10	19012	0.000993	0.002450	0.004205	0.007653	0.016236	0.016236
		0.001528	0.003412	0.004700	0.008416	0.022224	0.028520
		37.891	58.625	67.766	70.969	72.906	73.547

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

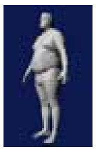
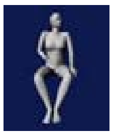
Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
11	21994	0.000465	0.005455	0.008881	0.009683	0.014576	0.022895
		0.001086	0.002134	0.004617	0.010499	0.042660	0.046877
		50.633	78.583	86.554	89.479	91.271	91.862
12	11224	0.005916	0.006803	0.008752	0.013763	0.017112	0.053291
		0.001319	0.002787	0.005350	0.010762	0.017468	0.037312
		23.103	35.430	41.450	43.072	44.574	45.195
13	27402	0.002954	0.002954	0.005334	0.010978	0.018783	0.043622
		0.001163	0.002322	0.004214	0.007249	0.020517	0.038733
		61.844	96.078	110.719	118.188	120.609	121.203
14	11378	0.001736	0.003207	0.005490	0.009510	0.034676	0.034676
		0.001518	0.004575	0.011178	0.022490	0.065200	0.081243
		23.469	39.109	42.250	43.516	44.578	44.984
15	11258	0.002190	0.004832	0.008484	0.017178	0.037969	0.053825
		0.002342	0.005748	0.012775	0.038460	0.058182	0.100492
		19.328	32.437	35.321	36.833	38.145	38.826
16	30450	0.004354	0.004431	0.011743	0.014006	0.014865	0.030845
		0.001525	0.003801	0.006686	0.010452	0.019276	0.035574
		46.407	72.875	84.532	90.340	92.623	93.504
17	31816	0.003263	0.004775	0.005560	0.009595	0.015434	0.021837
		0.001567	0.003189	0.006081	0.012974	0.025317	0.051791
		48.980	76.921	89.008	94.997	97.330	98.351

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

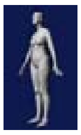
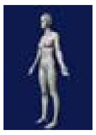
Model	Faces	50%	20%	10%	5%	2%	1%
18	30766	0.003344	0.007573	0.007573	0.008545	0.016463	0.022869
		0.002343	0.003291	0.006332	0.010417	0.033494	0.049290
		47.889	74.537	86.514	92.353	94.516	95.337
19	30950	0.002922	0.003350	0.005389	0.009624	0.023589	0.023589
		0.001022	0.002190	0.004442	0.009257	0.023499	0.040833
		46.537	71.953	83.470	89.749	92.203	93.164
20	31396	0.005243	0.005247	0.008305	0.008305	0.017013	0.017092
		0.001270	0.003418	0.006048	0.011358	0.023708	0.044669
		47.298	73.666	85.273	91.191	93.484	94.225
21	30396	0.000456	0.001545	0.002955	0.006206	0.012502	0.023335
		0.000828	0.001451	0.003182	0.005664	0.024215	0.036284
		52.185	78.583	86.554	89.479	91.271	91.862
22	30004	0.000458	0.001350	0.003480	0.004213	0.020042	0.018304
		0.000768	0.001642	0.003879	0.017423	0.088027	0.088027
		49.301	83.140	93.314	98.912	101.436	102.017
23	30074	0.000410	0.001318	0.003493	0.005569	0.011790	0.022425
		0.000702	0.001462	0.002988	0.005086	0.018029	0.029649
		56.712	111.330	122.086	125.851	129.086	130.478
24	30492	0.000488	0.001400	0.002798	0.005451	0.014162	0.024291
		0.000866	0.001612	0.003946	0.012693	0.089179	0.145632
		53.948	91.231	102.327	108.246	111.410	112.151

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
25	30170	0.000649	0.002587	0.002573	0.004101	0.007494	0.015958
		0.000679	0.001842	0.003073	0.005299	0.012456	0.021934
		58.000	97.625	109.094	113.250	116.281	117.484
26	30418	0.000649	0.001978	0.002908	0.005539	0.010608	0.021418
		0.000807	0.001753	0.003114	0.006784	0.022202	0.029708
		58.906	98.844	110.703	115.016	118.422	119.781
27	30274	0.002074	0.003909	0.006399	0.005692	0.017711	0.019446
		0.000964	0.002263	0.004075	0.006605	0.014685	0.037336
		58.672	99.094	110.735	115.047	118.578	120.016
28	30140	0.001300	0.002887	0.003115	0.016588	0.022814	0.030356
		0.000785	0.001405	0.002549	0.006865	0.021322	0.041434
		56.611	94.346	104.570	108.356	111.210	112.562
29	30254	0.000556	0.001631	0.003149	0.005770	0.020690	0.027292
		0.000793	0.002041	0.004016	0.006430	0.013867	0.048340
		51.594	87.376	98.311	102.127	104.891	106.073
30	30008	0.000318	0.001306	0.002369	*0.003500*	0.008099	0.015741
		0.000728	0.001254	0.002263	0.004986	0.012150	0.017032
		51.193	87.606	98.622	104.500	107.214	108.156
31	30454	0.000997	0.003755	0.003755	0.004534	0.019171	0.029141
		0.000911	0.001629	0.002686	0.006126	0.016485	0.029743
		54.919	92.553	104.100	108.075	111.310	112.832

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
32	29502	0.000564	0.001333	0.003219	0.007152	0.020644	0.025263
		0.000849	0.001395	0.002614	0.008315	0.015619	0.029611
		58.125	99.016	110.625	114.531	117.266	118.109
33	14698	0.000837	0.003046	0.006228	0.010887	0.022207	0.034847
		0.001014	0.002488	0.007117	0.014981	0.035364	0.077658
		22.693	38.796	42.301	44.674	46.236	46.767
34	19204	0.000635	0.002217	0.004939	0.008474	0.019095	0.022977
		0.001083	0.002254	0.005496	0.006917	0.022813	0.041774
		33.338	55.920	62.760	65.094	66.416	67.026
35	30268	0.001758	0.004544	0.005732	0.005732	0.018311	0.017410
		0.000886	0.002109	0.002857	0.005775	0.015705	0.033037
		49.571	83.780	93.735	97.280	99.783	100.895
36	18152	0.000598	0.001893	0.004640	0.008422	0.016085	0.022346
		0.000975	0.001929	0.003740	0.009081	0.020196	0.032004
		25.306	41.700	49.231	51.975	53.898	54.588
37	12540	0.001993	0.003772	0.005814	0.010357	0.032548	0.052556
		0.002306	0.003542	0.006307	0.014257	0.043726	0.171060
		17.756	29.152	34.309	36.222	37.203	37.784
38	30128	0.000782	0.004475	0.003525	0.005931	0.022479	0.024220
		0.000758	0.002594	0.005697	0.030728	0.070070	0.069017
		48.550	82.639	93.254	99.253	101.206	101.576
39	30322	0.006473	0.006441	0.006382	0.006335	0.009185	0.022904
		0.000769	0.003540	0.007273	0.014674	0.049402	**0.357970**
		51.965	88.367	99.153	102.948	105.311	106.153

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

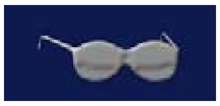
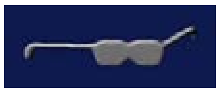
Model	Faces	50%	20%	10%	5%	2%	1%
40	30326	0.000487	0.001761	0.002544	0.005596	0.009671	0.019241
		0.000753	0.001768	0.002691	0.004608	0.012667	0.025731
		49.111	82.308	88.968	94.806	97.490	98.321
41	14028	0.000520	0.002056	0.003508	0.005245	0.013232	0.017964
		0.000735	0.002029	0.005805	0.012352	0.016966	0.029383
		23.023	38.936	42.431	44.474	45.616	46.347
42	8324	0.000423	0.001996	0.005759	0.007493	0.010440	0.071092
		0.001645	0.002367	0.007110	0.019858	0.070725	0.102936
		13.710	22.302	25.316	26.859	28.201	28.651
43	6824	0.001373	0.002217	0.006380	0.007938	0.012653	0.043182
		0.000990	0.003806	0.008591	0.015301	0.040651	0.088274
		11.426	18.186	20.279	21.231	22.042	22.532
44	12572	0.000631	0.002280	0.003378	0.005309	0.014003	0.018842
		0.001199	0.002089	0.006364	0.015581	0.032212	0.062266
		20.900	35.321	38.455	40.348	41.380	42.161
45	10794	0.000792	0.002473	0.004751	0.005987	0.032257	0.134932
		0.000860	0.002863	0.007161	0.013574	0.030949	0.076616
		17.155	29.432	32.146	33.869	34.800	35.361
46	4784	0.004670	0.004868	0.007586	0.007546	0.016756	0.023167
		0.001160	0.002907	0.008529	0.015014	0.025008	0.040615
		8.332	12.117	13.860	14.611	15.342	15.863
47	4052	0.001509	0.006504	0.010646	0.013609	0.019756	0.023448
		0.001208	0.004181	0.009266	0.010380	0.025902	0.050801
		7.571	11.076	12.768	13.620	14.401	14.761

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

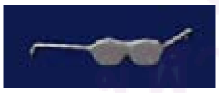
Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
48	10348	0.000487	0.001720	0.005243	0.008773	0.015047	0.021492
		0.000814	0.002212	0.005572	0.013354	0.121845	**0.487676**
		18.006	30.304	32.947	34.259	35.611	36.142
49	6220	0.000860	0.002930	0.005279	0.008233	0.021276	0.028307
		0.001435	0.002773	0.006283	0.012711	0.042175	0.072208
		11.196	17.575	19.598	20.459	21.301	21.781
50	14836	0.000540	0.002004	0.003598	0.007470	0.025911	0.017804
		0.000970	0.002289	0.006531	0.011449	0.023909	0.046701
		24.625	41.349	45.035	47.458	48.830	49.361
51	5712	0.001150	0.003325	0.004980	0.011593	0.024247	0.141063
		0.001045	0.002857	0.007269	0.019807	0.035696	0.082072
		9.664	15.262	17.145	18.326	19.158	19.708
52	3104	0.001364	0.003055	0.006005	0.022856	**0.175207**	0.175207
		0.001473	0.004682	0.010132	0.034419	0.038693	0.077308
		4.376	8.202	9.133	9.904	10.585	11.166
53	4988	0.001141	0.003475	0.006074	0.017029	0.019455	0.022682
		0.001278	0.003909	0.009662	0.009860	0.031981	0.045562
		8.923	13.580	14.591	15.412	16.354	17.004
54	14810	0.000492	0.002382	0.003619	0.006366	0.012164	0.018160
		0.000731	0.002057	0.006884	0.017011	0.018499	0.053298
		24.615	41.580	45.535	47.909	49.511	50.172
55	5696	0.001218	0.004088	0.006784	0.009452	0.014404	0.116381
		0.001367	0.003615	0.008795	0.014332	0.043458	0.054060
		10.085	15.562	17.475	18.266	19.128	19.798

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
56	17538	0.000499	0.003140	0.006758	0.007169	0.017110	0.032494
		0.000853	0.001809	0.004356	0.007388	0.013144	0.024416
		28.171	49.351	55.750	57.883	59.165	59.856
57	14822	0.000454	0.002248	0.003759	0.008865	0.013459	0.019320
		0.000762	0.002628	0.005564	0.010502	0.018758	0.043901
		24.535	41.510	45.385	47.839	49.231	49.802
58	3710	0.000986	0.004733	0.005603	0.012400	0.021741	0.031148
		0.001202	0.007163	0.009217	0.016944	0.025045	0.057074
		6.589	9.083	10.255	11.176	11.907	12.218
59	5786	0.000955	0.003010	0.004133	0.009255	0.028841	0.028858
		0.001422	0.002516	0.005476	0.013215	0.032966	0.070953
		10.896	16.464	18.326	19.077	19.829	20.309
60	4746	0.000775	0.003386	0.005035	0.011027	0.012460	0.023706
		0.001300	0.004287	0.006844	0.012278	0.050447	0.050530
		8.112	11.777	13.690	14.571	15.312	15.683
61	10796	0.000536	0.002478	0.003989	0.005720	0.016546	0.024715
		0.000936	0.002663	0.005099	0.007859	0.017481	0.048546
		19.418	33.078	36.202	38.355	39.817	40.338
62	11234	0.000637	0.002391	0.004241	0.006018	0.012813	0.027283
		0.001021	0.001955	0.004364	0.008603	0.018469	0.034241
		18.647	31.305	33.939	35.801	36.633	37.073
63	11034	0.000361	0.001797	0.003470	0.007774	0.015434	0.037876
		0.000921	0.002081	0.004356	0.008727	0.018563	**0.298004**
		17.425	27.670	32.146	33.358	34.450	34.880

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

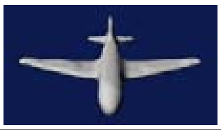
Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
64	13590	0.000704	0.002229	0.003605	0.006881	0.015574	0.022538
		0.001313	0.002079	0.004485	0.008959	0.016906	0.037636
		22.472	37.935	41.229	43.352	44.454	45.035
65	12892	0.000525	0.002151	0.003973	0.006451	0.014462	0.016017
		0.000870	0.001727	0.004444	0.006799	0.015688	0.019473
		21.781	36.663	39.927	41.920	43.042	43.623
66	15474	0.000161	0.001357	0.003329	0.005211	0.008953	0.020068
		0.000562	0.001708	0.003668	0.006556	0.015426	0.022410
		24.425	43.212	49.291	51.384	52.435	52.876
67	15298	0.000338	0.001641	0.002880	0.005286	0.012643	0.026596
		0.000636	0.001847	0.003832	0.007218	0.012380	0.023534
		24.005	42.421	48.289	50.182	51.204	51.644
68	11162	0.000575	0.002567	0.003732	0.007637	0.013518	0.026025
		0.000823	0.002225	0.004088	0.008769	0.015798	0.025182
		18.687	31.235	33.899	35.691	36.513	36.943
69	13398	0.000337	0.001580	0.002760	0.005664	0.014374	0.018674
		0.000645	0.001707	0.004439	0.009875	0.022968	0.122371
		22.032	37.844	41.480	43.733	44.955	45.505
70	11258	0.000347	0.002259	0.003545	0.007512	0.034493	0.043923
		0.000950	0.001903	0.003970	0.007964	0.019088	0.070189
		18.056	31.285	34.259	36.052	37.023	37.614
71	11694	0.000450	0.001782	0.003654	0.006769	0.009775	0.022950
		0.000866	0.001814	0.004290	0.008492	0.015972	0.033649
		19.598	32.597	35.461	37.224	38.295	38.776

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

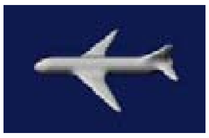
Model	Faces	50%	20%	10%	5%	2%	1%
72	10452	0.000477	0.002253	0.004287	0.009333	0.018166	0.017433
		0.001067	0.002696	0.004018	0.009133	0.026254	0.037230
		16.734	28.571	31.185	32.306	33.438	33.909
73	10214	0.000611	0.002102	0.004337	0.008375	0.012048	0.019892
		0.001021	0.002304	0.004553	0.008757	0.017561	0.024618
		16.614	28.271	30.734	31.816	32.867	33.298
74	10084	0.000618	0.001861	0.003904	0.007359	0.011333	0.022221
		0.001068	0.002731	0.004934	0.013017	0.020280	0.047445
		16.674	28.461	30.975	32.176	33.338	33.979
75	17354	0.000441	0.001632	0.003649	0.004483	0.009898	0.017749
		0.000896	0.001551	0.003459	0.005426	0.012140	0.019228
		28.831	48.770	54.949	57.102	58.154	58.744
76	11842	0.000415	0.001625	0.004932	0.005208	0.012860	0.018548
		0.000735	0.002132	0.004273	0.008284	0.018641	0.023337
		17.715	31.385	34.470	36.032	37.003	37.454
77	13170	0.000468	0.001776	0.002842	0.005595	0.014212	0.015560
		0.000958	0.001622	0.004186	0.007102	0.015671	0.022026
		21.531	36.472	39.817	41.890	42.772	43.392
78	14936	0.000333	0.001332	0.002444	0.003877	0.016021	0.022129
		0.000998	0.002084	0.003241	0.005507	0.015677	0.026219
		24.215	41.159	44.905	47.378	48.800	49.251
79	13374	0.000349	0.001748	0.004105	0.005774	0.011998	0.020762
		0.000717	0.001657	0.003944	0.007338	0.014342	0.024275
		21.801	37.143	40.478	42.521	43.553	44.023

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

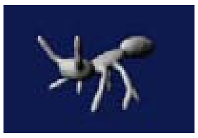
Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
80	14698	0.000360	0.001685	0.003259	0.005551	0.015376	0.018567
		0.000644	0.001758	0.003027	0.006742	0.013420	0.024816
		23.283	40.508	44.564	47.138	48.750	49.291
81	12736	0.001768	0.003946	0.005353	0.011366	0.027569	0.034290
		0.001463	0.004588	0.013379	0.022718	0.035793	0.065673
		21.721	37.143	41.029	43.392	45.045	45.816
82	16594	0.000874	0.003428	0.006018	0.013026	0.029155	0.027886
		0.001181	0.002999	0.006998	0.014243	0.034852	0.059523
		28.000	46.958	51.234	53.898	55.650	56.371
83	12748	0.001815	0.003789	0.006423	0.009955	0.022021	0.096273
		0.001476	0.004585	0.008928	0.018279	0.039005	0.056489
		21.241	35.661	38.856	40.719	41.860	42.541
84	13696	0.001257	0.003380	0.007003	0.010610	0.026811	0.037560
		0.001225	0.003312	0.007413	0.013982	0.037606	0.062843
		22.613	38.285	41.660	43.733	45.215	45.776
85	16772	0.000956	0.003230	0.006266	0.015573	0.029114	0.030009
		0.001415	0.003340	0.008854	0.019719	0.039061	0.063196
		28.010	47.108	51.364	54.108	55.810	56.621
86	14072	0.001048	0.003933	0.007140	0.015672	0.023446	0.040536
		0.001450	0.003649	0.009767	0.016791	0.041171	0.066506
		23.504	39.367	42.892	45.105	46.647	47.418

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

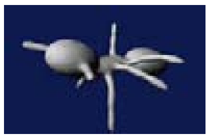
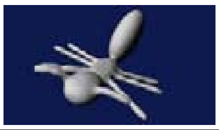
Model	Faces	50%	20%	10%	5%	2%	1%
87	14936	0.001758	0.002775	0.005684	0.009261	0.024654	0.029749
		0.001440	0.003130	0.008026	0.015632	0.031728	0.061128
		24.285	41.380	45.305	47.909	49.611	50.202
88	16416	0.000939	0.003094	0.005546	0.008851	0.024964	0.023400
		0.001213	0.002746	0.005726	0.018812	0.029473	0.048502
		27.309	46.036	50.192	52.866	55.099	55.800
89	15304	0.001301	0.003231	0.006483	0.013723	0.023992	0.031391
		0.001622	0.003803	0.008743	0.014672	0.043997	0.065579
		25.647	43.162	47.058	49.621	51.364	52.005
90	15750	0.001062	0.003403	0.007992	0.010002	0.019083	0.026732
		0.001323	0.003658	0.008591	0.019088	0.050614	0.072136
		25.957	43.693	47.639	50.292	51.995	52.646
91	17276	0.001094	0.003254	0.006445	0.013678	0.021187	0.028266
		0.001305	0.002815	0.006861	0.015804	0.039400	0.062954
		28.511	48.299	54.298	56.421	57.553	58.414
92	11730	0.001916	0.003606	0.008933	0.012003	0.025680	0.076213
		0.001700	0.003905	0.009087	0.020744	0.049586	0.093215
		18.807	32.497	35.491	37.364	38.636	39.297
93	16432	0.000809	0.003332	0.006172	0.009387	0.022010	0.034946
		0.001769	0.002725	0.008535	0.012775	0.047612	0.105190
		27.089	45.977	51.754	53.086	55.219	56.111

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

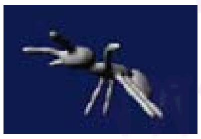
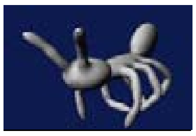
Model	Faces	50%	20%	10%	5%	2%	1%
94	14468	0.001683	0.003008	0.006121	0.010479	0.018464	0.030114
		0.001557	0.003000	0.006021	0.014104	0.033916	0.044007
		24.165	40.308	40.063	46.206	47.689	48.279
95	14706	0.000984	0.003604	0.005690	0.010578	0.022865	0.039574
		0.001123	0.003442	0.008047	0.018715	0.037867	0.057851
		23.654	40.648	44.264	46.767	48.279	48.750
96	17004	0.001007	0.004228	0.005593	0.009940	0.020852	0.042139
		0.001480	0.003030	0.006823	0.014600	0.030414	0.072939
		28.751	47.969	52.025	54.619	56.401	57.052
97	16330	0.001366	0.004283	0.006576	0.017716	0.024356	0.035479
		0.001471	0.003500	0.008257	0.018087	0.038154	0.066226
		27.600	46.146	50.402	53.056	54.949	55.800
98	12304	0.001308	0.004877	0.005923	0.009647	0.020492	0.040508
		0.001767	0.003513	0.010243	0.021207	0.052989	0.060210
		20.660	34.820	37.955	39.767	41.129	41.790
99	13758	0.001167	0.003056	0.006310	0.010305	0.020295	0.032677
		0.001251	0.003313	0.006999	0.022022	0.032188	0.056484
		23.624	38.986	42.231	44.374	45.676	46.327
100	16124	0.001133	0.004149	0.005076	0.009841	0.019381	0.039986
		0.001394	0.003043	0.009372	0.015122	0.037247	0.094443
		32.813	55.703	61.000	64.063	65.844	66.438
101	16998	0.000054	0.001452	0.002917	0.007594	0.016074	0.020075
		0.000358	0.001688	0.004069	0.007071	0.015602	0.024933
		35.891	59.250	66.953	70.047	71.172	71.781

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

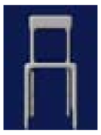
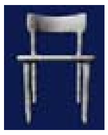
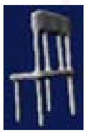
Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
102	31456	0.016001	0.016001	0.010288	0.010775	0.017110	0.025999
		0.000855	0.001838	0.003465	0.006740	0.014810	0.020548
		66.171	112.063	125.672	132.719	135.250	136.016
103	19304	0.000469	0.002334	0.009989	0.009989	0.012717	0.023701
		0.001085	0.002174	0.005711	0.009971	0.022975	0.079994
		39.922	67.625	76.500	79.5625	81.266	81.891
104	17306	0.000557	0.000820	0.003351	0.006207	0.018272	0.025251
		0.000332	0.002780	0.006833	0.021807	0.039594	0.226061
		37.656	56.453	65.500	68.563	70.141	70.547
105	18530	0.000740	0.001484	0.003975	0.006845	0.017181	0.025543
		0.000695	0.002568	0.008725	0.031098	**0.303668**	N/A*
		38.641	64.797	70.953	75.984	76.828	N/A*
106	28104	0.008109	0.008610	0.008563	0.008610	0.011811	0.028546
		0.000709	0.001910	0.003809	0.008179	0.023598	0.087128
		69.156	99.234	116.297	120.313	123.031	123.797
107	22854	0.000529	0.002546	0.004147	0.008967	0.017935	0.023606
		0.001108	0.002199	0.004730	0.007809	0.027350	0.031460
		46.828	80.469	90.828	94.578	97.031	97.953
108	21008	0.000634	0.002520	0.004712	0.008372	0.016833	0.023065
		0.001158	0.002026	0.004225	0.008428	0.018176	0.027522
		42.982	73.886	83.410	87.345	90.230	91.862
109	20602	0.000082	0.001549	0.004268	0.007047	0.014546	0.044515
		0.000575	0.001778	0.007350	0.011116	0.043881	0.110542
		48.266	80.031	89.625	93.031	94.953	95.422

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

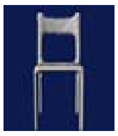
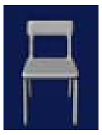
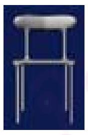
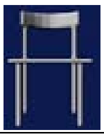
Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
110	16920	0.008116	0.008230	0.008646	0.008980	0.019305	0.045441
		0.001799	0.003104	0.010567	0.031547	0.128419	N/A*
		34.797	62.672	71.281	74.093	75.156	N/A*
111	16100	0.008094	0.008093	0.008093	0.008433	0.015287	0.024168
		0.001798	0.003412	0.009973	**0.110990**	0.137811	0.137811
		38.344	62.891	70.203	71.500	73.047	73.219
112	27256	0.006877	0.010115	0.007329	0.010444	0.024986	0.024986
		0.000735	0.001607	0.003457	0.009775	0.054521	**0.370797**
		66.719	108.922	122.609	129.734	132.344	132.750
113	21500	0.003610	0.003613	0.003610	0.008049	0.013139	0.020665
		0.001627	0.001855	0.005113	0.023223	0.148185	0.114980
		46.922	79.688	90.656	96.125	97.438	97.469
114	11588	0.000263	0.001891	0.065273	0.073310	0.073310	0.170187
		0.001267	0.004398	0.017078	0.050990	**0.309078**	**0.361201**
		23.109	39.656	43.188	45.094	45.953	46.188
115	20086	0.002028	0.007305	0.010736	0.005455	0.012650	0.018202
		0.001628	0.001628	0.006733	0.054788	0.119107	0.119107
		45.969	77.375	87.781	91.875	94.281	94.406
116	26926	0.026033	0.025989	0.026157	0.026168	0.026266	0.026522
		0.000182	0.001145	0.004168	0.015477	0.087511	0.173991
		61.172	97.703	110.203	116.297	117.578	118.047
117	28744	0.021494	0.022012	0.021923	0.021833	0.022277	0.022138
		*0.000000*	0.001608	0.004763	0.012955	0.090454	0.121110
		77.296	116.500	129.969	136.781	138.797	139.157

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

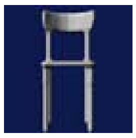
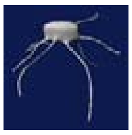
Model	Faces	50%	20%	10%	5%	2%	1%
118	18306	0.001013	0.003127	0.007241	0.009318	0.015834	0.021306
		0.000656	0.002372	0.004959	0.010851	0.022189	0.122442
		38.812	64.578	72.750	75.531	76.922	77.750
119	24652	0.000655	0.010680	0.020890	0.009022	0.016899	0.023350
		0.001614	0.001614	0.004099	0.019477	0.075763	0.076477
		61.250	93.953	105.625	109.969	112.484	112.750
120	20242	0.012452	0.017438	0.024766	0.024642	0.025298	0.025470
		0.000135	0.001709	0.005199	0.035487	0.148873	0.214718
		48.797	77.188	85.875	88.828	89.875	90.125
121	11888	0.001269	0.003310	0.006232	0.013082	0.018419	0.025329
		0.001723	0.003308	0.009335	0.016785	0.035186	0.054382
		23.484	38.896	42.571	45.065	47.488	48.700
122	14498	0.001040	0.003063	0.005406	0.009556	0.029307	0.026081
		0.001181	0.002663	0.006204	0.013997	0.025692	0.041365
		29.313	49.234	53.297	55.844	57.422	57.938
123	30982	0.000546	0.001842	0.004118	0.008383	0.020469	0.025811
		0.001069	0.001726	0.005123	0.011176	0.029328	0.045709
		60.938	103.516	116.266	120.922	124.766	125.484
124	6198	0.001465	0.002916	0.005836	0.008556	0.064251	**0.258244**
		0.002545	0.009326	0.021552	0.024266	0.068673	0.206997
		12.922	19.547	21.734	22.469	23.171	23.422
125	2682	0.002958	0.027280	0.065240	**0.168817**	**0.198215**	**0.214263**
		0.008017	**0.038168**	**0.067967**	**0.091343**	**0.208560**	0.208560
		4.484	7.031	8.031	8.625	9.141	9.219

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

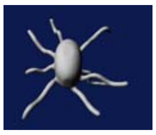
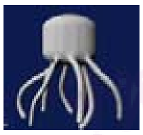
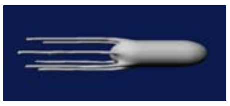
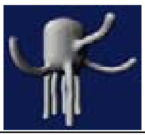
Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
126	12646	0.001289	0.002855	0.005444	0.010276	0.016837	0.021297
		0.001489	0.004252	0.009041	0.019612	0.034690	0.116973
		27.750	44.781	48.188	50.297	51.406	51.828
127	23808	0.000803	0.002549	0.005060	0.007306	0.024910	0.025467
		0.000996	0.002499	0.004262	0.012666	0.028093	0.061302
		47.938	83.719	93.593	96.656	98.797	99.547
128	12324	0.001234	0.003333	0.006754	0.014397	0.018960	0.104058
		0.001415	0.003935	0.008298	0.021164	0.044172	0.149055
		25.875	42.438	45.797	47.766	48.828	49.281
129	12284	0.001302	0.003537	0.005650	0.010671	0.017355	0.107116
		0.001189	0.005006	0.008548	0.023193	0.048438	0.215950
		24.109	40.469	44.031	46.047	47.078	47.531
130	15490	0.000956	0.003399	0.005526	0.008003	0.022037	0.030966
		0.001175	0.002980	0.008531	0.015678	0.034066	0.063405
		30.047	50.484	55.250	58.719	61.203	61.875
131	22098	0.000722	0.002950	0.003818	0.007296	0.014810	0.030555
		0.001141	0.002175	0.004784	0.009780	0.039497	0.039497
		48.547	81.813	92.328	95.781	98.172	99.031
132	15620	0.000351	0.001860	0.006173	0.007759	0.018602	0.020759
		0.001272	0.004459	0.018672	0.030110	0.065135	0.143948
		29.640	47.453	55.031	57.531	58.922	59.188
133	20932	0.000512	0.001867	0.003633	0.007893	0.022999	0.021785
		0.001223	0.002546	0.005285	0.014902	0.032560	0.089986
		39.266	66.594	74.563	77.250	78.609	79.391

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

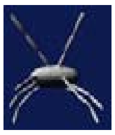
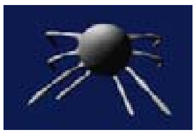
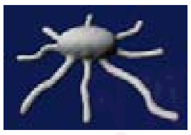
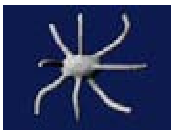
Model	Faces	50%	20%	10%	5%	2%	1%
134	21860	0.000931	0.003248	0.005398	0.011213	0.025547	0.042928
		0.001404	0.002643	0.005501	0.012912	0.034867	0.054317
		46.337	78.122	86.164	90.881	93.885	95.257
135	30398	0.000694	0.002377	0.004172	0.007528	0.014805	0.026230
		0.001245	0.002750	0.005560	0.010158	0.022819	0.037079
		56.938	95.422	106.578	110.578	113.344	114.469
136	28248	0.000759	0.002190	0.004596	0.005361	0.024108	0.021799
		0.000973	0.002186	0.003474	0.007542	0.019627	0.037228
		62.703	105.969	118.4375	122.906	126.000	127.094
137	18244	0.001415	0.002278	0.004851	0.009644	0.012918	0.018238
		0.001471	0.003009	0.009870	0.019497	0.033424	0.080220
		33.156	61.156	66.203	70.422	71.438	72.094
138	19184	0.000971	0.004039	0.005009	0.009545	0.014825	0.028533
		0.001170	0.002796	0.008277	0.020057	0.047324	0.147565
		34.719	60.063	67.500	70.031	71.359	72.109
139	16030	0.000952	0.003292	0.006956	0.009619	0.018169	0.026486
		0.001631	0.002817	0.006800	0.011664	0.027451	0.040413
		30.344	51.594	56.031	58.906	60.578	61.125
140	18396	0.001003	0.002502	0.003981	0.008026	0.013646	0.022397
		0.001080	0.002310	0.004719	0.014409	0.022499	0.048058
		35.187	58.906	63.875	67.078	69.156	69.844
141	27848	0.011917	0.023325	0.023006	0.023205	0.023405	0.023343
		0.000000	0.000798	0.002153	0.004322	0.018364	0.021316
		62.859	98.843	111.344	116.484	119.813	120.813

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont’d)

Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
142	19600	0.000012	0.000998	0.003429	0.004899	0.019287	0.019311
		0.000011	0.001730	0.003461	0.006788	0.014599	0.032945
		39.531	67.625	75.438	78.094	79.375	80.188
143	18536	0.000157	0.001104	0.001679	0.004715	0.009497	0.014222
		0.000000	0.001388	0.002033	0.006393	0.013693	0.036156
		46.391	77.219	86.093	89.281	91.156	91.578
144	30160	0.000132	0.021038	0.020640	0.020492	0.020930	0.020624
		0.000000	0.000099	0.001639	0.002253	0.009003	*0.012654*
		78.531	120.984	133.516	139.891	142.172	142.688
145	26408	0.002728	0.005576	0.008912	0.012276	0.015450	0.018838
		0.000195	0.001404	0.002228	0.005238	0.010691	0.021375
		45.438	81.672	94.531	99.453	102.234	103.078
146	27424	0.005020	0.013443	0.013443	0.029930	0.029930	0.029930
		0.001241	0.002476	0.003997	0.006116	0.016733	0.023292
		57.643	93.314	104.130	109.748	112.091	113.043
147	27406	0.000112	0.008788	0.011605	0.008817	0.017714	0.020873
		0.000000	0.001143	0.002366	0.003224	0.009957	0.021564
		54.408	82.929	93.554	99.082	101.145	102.307
148	26746	0.008595	0.020942	0.021294	0.020954	0.021342	0.021814
		0.000008	0.001177	0.002400	0.005750	0.013848	0.017740
		49.101	77.692	88.377	93.925	96.238	97.060

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

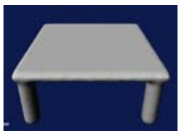
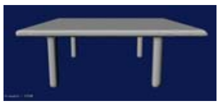
Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
149	27172	0.001826	0.020588	0.020633	0.020312	0.021505	0.020868
		0.000000	0.001273	0.001645	0.004020	0.008213	0.020691
		53.136	82.689	92.884	98.502	100.505	101.856
150	27154	0.020048	0.019793	0.020133	0.019999	0.019858	0.021119
		0.000571	0.001562	0.003153	0.005195	0.009667	0.019267
		49.641	80.215	90.029	93.895	96.809	98.111
151	25388	*0.000004*	0.003683	0.004407	0.007624	0.013172	0.026694
		0.000002	0.001568	0.002214	0.004966	0.009630	0.019507
		48.688	82.844	95.750	100.156	103.156	103.875
152	21082	0.000033	0.000327	*0.000768*	*0.001473*	0.007465	*0.010454*
		0.001715	0.001715	0.005684	0.025529	0.044077	0.044203
		57.372	84.171	95.097	99.263	101.786	102.007
153	27158	0.015292	0.020647	0.020473	0.019622	0.020740	0.021014
		0.000000	0.000662	0.001400	0.003152	0.011615	0.019146
		55.009	82.539	92.313	97.711	99.763	100.855
154	27854	0.000479	0.001906	0.003186	0.004949	0.012614	0.030213
		0.000996	0.001749	0.002548	0.004556	0.009287	0.017418
		43.262	74.417	83.931	87.686	90.630	91.832
155	28764	0.022793	0.023286	0.023064	0.023228	0.023236	0.023402
		0.000001	0.000975	0.002147	0.004241	0.015284	0.020961
		53.898	90.430	101.977	108.186	110.549	111.500
156	20196	*0.000002*	0.002550	0.002799	0.004907	0.008088	0.015609
		0.000000	0.001399	0.002556	0.004341	0.015548	0.024295
		53.166	81.237	88.517	91.401	93.545	94.316

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

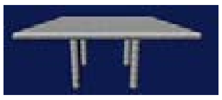
Model	Faces	50%	20%	10%	5%	2%	1%
157	27762	0.018161	0.018772	0.018759	0.018763	0.018789	0.018856
		0.000000	0.001366	0.001721	0.003845	0.014404	0.015337
		54.739	84.992	95.718	101.566	104.050	105.011
158	29170	0.000053	0.004826	0.007494	0.007494	0.013642	0.020611
		0.000131	0.001173	0.002273	0.008234	0.012307	0.020828
		85.523	124.229	135.805	141.994	144.297	145.359
159	27856	0.022750	0.022441	0.022230	0.022400	0.022746	0.022443
		0.000010	0.001314	0.002351	0.003765	0.008710	0.021093
		51.664	81.067	91.712	97.510	99.814	100.995
160	20160	0.015160	0.015160	0.015160	0.015227	0.016493	0.016318
		0.004532	0.004532	0.005301	0.048770	0.062318	N/A[*]
		55.430	75.939	85.863	89.599	90.831	N/A[*]
161	27648	0.000636	0.001728	0.003859	0.006812	0.023064	0.028744
		0.001057	0.002250	0.003568	0.006740	0.018576	0.030684
		66.976	102.497	112.121	115.586	118.210	119.452
162	20188	0.000710	0.002006	0.003746	0.007974	0.017460	0.030127
		0.001168	0.002142	0.003645	0.007297	0.016744	0.027632
		33.027	56.061	62.900	65.454	67.347	68.438
163	21758	0.000631	0.001755	0.003463	0.006452	0.027902	0.028889
		0.001117	0.001786	0.003837	0.006901	0.019222	0.029887
		36.202	61.559	69.149	72.004	74.387	75.428
164	29014	0.000548	0.001559	0.003829	0.010786	0.012553	0.021372
		0.001125	0.002062	0.002582	0.006002	0.016495	0.029671
		48.350	82.669	92.673	96.689	99.804	101.135

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

Model	Faces	50%	20%	10%	5%	2%	1%
165	20462	0.000743	0.002078	0.003873	0.008466	0.017559	0.026919
		0.001093	0.001774	0.003487	0.007450	0.017885	0.042395
		34.049	57.933	65.134	67.848	69.820	70.982
166	22176	0.000615	0.002108	0.003884	0.007763	0.014936	0.026147
		0.000953	0.002001	0.003054	0.006148	0.015093	0.023279
		36.843	62.300	70.051	72.975	75.348	76.460
167	25290	0.000555	0.002153	0.003955	0.006353	0.014970	0.024094
		0.000863	0.001798	0.002885	0.006081	0.013832	0.026604
		42.291	71.362	80.075	83.360	85.713	86.805
168	21500	0.000621	0.001902	0.004184	0.007267	0.019106	0.025951
		0.001060	0.002349	0.003585	0.006920	0.015027	0.026245
		35.571	60.807	68.388	71.323	73.766	74.788
169	25118	0.000527	0.002087	0.002996	0.006015	0.011797	0.019042
		0.000858	0.001805	0.003984	0.005673	0.012990	0.020789
		42.281	71.939	79.965	83.180	85.563	86.795
170	24996	0.000595	0.001828	0.003050	0.005619	0.011755	0.021255
		0.000913	0.001889	0.002514	0.004993	0.014151	0.023564
		41.329	70.541	79.094	82.308	84.572	85.883
171	29806	0.000525	0.002130	0.003725	0.007122	0.010932	0.026198
		0.000819	0.001342	0.002617	0.005998	0.018948	0.026481
		52.505	89.549	100.525	104.630	107.845	109.087
172	20278	0.000691	0.002110	0.004186	0.008053	0.017056	0.028611
		0.001363	0.001806	0.003614	0.007569	0.016960	0.030721
		33.769	57.413	64.373	67.026	69.019	70.121

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

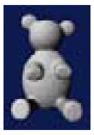
Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
173	27730	0.000600	0.002046	0.003545	0.007540	0.013687	0.025415
		0.001007	0.001555	0.003143	0.008311	0.017104	0.032521
		46.206	77.832	87.696	91.502	94.466	95.678
174	25658	0.000495	0.002447	0.002766	0.005795	0.011725	0.023076
		0.000830	0.001488	0.002874	0.005839	0.014440	0.036838
		43.422	72.865	81.697	85.072	87.596	88.567
175	18956	0.000663	0.002354	0.004413	0.008195	0.019642	0.033192
		0.001108	0.001924	0.003426	0.007903	0.019494	0.032622
		31.465	53.317	59.966	62.400	63.962	64.924
176	24996	0.000601	0.001846	0.004105	0.006057	0.014522	0.028947
		0.000936	0.002092	0.002997	0.008084	0.017783	0.040781
		43.202	72.114	81.167	84.522	87.155	88.227
177	25082	0.000563	0.001977	0.004002	0.008223	0.012643	0.030202
		0.001208	0.001827	0.003376	0.005564	0.018199	0.036952
		42.301	70.832	79.574	82.949	85.613	86.734
178	24346	0.000701	0.002074	0.003358	0.006445	0.013679	0.022537
		0.000943	0.002187	0.003171	0.006664	0.021391	0.032303
		41.159	69.520	78.072	81.327	83.600	84.532
179	26644	0.000536	0.001804	0.003299	0.006095	0.015011	0.026472
		0.001040	0.001415	0.003402	0.005716	0.021106	0.036033
		45.405	75.899	85.152	88.667	91.521	92.583
180	19092	0.000640	0.002105	0.004447	0.010443	0.014069	0.032749
		0.001084	0.001879	0.003395	0.006631	0.024308	0.042316
		31.415	53.597	58.484	62.490	64.162	65.044

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

Model	Faces	50%	20%	10%	5%	2%	1%
181	14480	0.000847	0.002703	0.005576	0.009162	0.017603	0.021623
		0.001321	0.002688	0.004843	0.009168	0.026919	0.033829
		27.844	47.328	51.391	54.047	56.141	57.141
182	20562	0.000690	0.002180	0.003875	0.007787	0.015905	0.026539
		0.001202	0.002377	0.003573	0.011656	0.021094	0.041328
		33.969	56.702	63.531	66.025	67.888	69.300
183	22822	0.000674	0.001886	0.003731	0.006245	0.015271	0.019417
		0.001001	0.001956	0.003903	0.005649	0.015379	0.033268
		35.030	60.908	68.258	71.052	73.376	74.287
184	9366	0.001635	0.004604	0.011158	0.011603	0.022244	0.042038
		0.002357	0.004139	0.010296	0.021103	0.041762	0.096615
		15.212	24.185	27.059	28.821	30.304	31.035
185	4974	0.003347	0.012195	0.011969	0.024789	0.038689	0.048276
		0.003322	0.011239	0.019011	0.053520	0.052015	0.117778
		8.412	12.358	14.040	15.072	15.933	16.313
186	13210	0.001034	0.002662	0.004959	0.008545	0.017520	0.022013
		0.001321	0.003180	0.005150	0.016871	0.028135	0.047566
		21.361	35.922	39.046	41.109	42.701	43.703
187	15256	0.000770	0.002568	0.004900	0.009491	0.016272	0.021524
		0.001322	0.002197	0.004818	0.008729	0.030209	0.042906
		25.867	42.822	46.627	49.251	51.103	52.005
188	17202	0.000766	0.002003	0.005508	0.005959	0.017664	0.023116
		0.001067	0.002087	0.004319	0.007243	0.021078	0.027301
		28.381	47.528	51.885	54.889	57.362	58.244

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

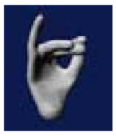
Model	Faces	50%	20%	10%	5%	2%	1%
189	28164	0.000531	0.001989	0.004548	0.005941	0.012083	0.018721
		0.000740	0.001607	0.002621	0.004890	0.012421	0.022562
		47.969	79.054	88.057	91.702	94.326	95.467
190	14996	0.000759	0.002827	0.005059	0.009783	0.018181	0.025621
		0.001118	0.003084	0.003617	0.008577	0.023913	0.032517
		24.996	42.782	47.038	50.002	50.125	53.196
191	27528	0.000566	0.002538	0.003708	0.005820	0.013365	0.019965
		0.000960	0.002136	0.004479	0.005807	0.023554	0.028798
		56.722	96.819	111.000	114.995	119.061	120.223
192	13978	0.000780	0.002277	0.005283	0.009351	0.021669	0.029908
		0.001042	0.002130	0.004326	0.010849	0.018533	0.030146
		23.544	39.387	43.012	45.265	46.907	47.779
193	14624	0.000840	0.003319	0.005315	0.010321	0.026369	0.039448
		0.001335	0.003462	0.005807	0.012249	0.023823	0.036995
		24.605	41.620	45.726	48.440	50.262	51.124
194	16938	0.000788	0.002514	0.005236	0.009118	0.018234	0.029507
		0.001125	0.002834	0.004093	0.010470	0.020766	0.051075
		28.942	47.809	52.315	55.320	57.503	58.594
195	17044	0.000879	0.002936	0.004307	0.010113	0.025604	0.030817
		0.001159	0.002593	0.004826	0.008401	0.030715	0.039573
		29.192	48.650	53.186	56.161	58.364	59.445
196	14220	0.000904	0.002824	0.005694	0.010933	0.018607	0.032680
		0.001321	0.003028	0.004101	0.009148	0.014683	0.051775
		23.564	39.907	43.773	46.507	48.550	49.311

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

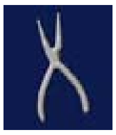
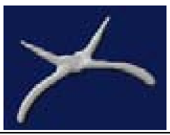
Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
197	15102	0.000820	0.002973	0.007323	0.014849	0.027483	0.034076
		0.001249	0.002289	0.005178	0.009080	0.031595	0.039703
		25.717	43.202	47.388	50.202	52.205	53.006
198	19124	0.001008	0.001766	0.003787	0.005189	0.015194	0.021681
		0.001074	0.001752	0.004358	0.008271	0.014003	0.034099
		30.975	52.025	56.792	59.996	62.229	63.221
199	17290	0.001000	0.003460	0.005242	0.009021	0.018667	0.034023
		0.001718	0.002496	0.006195	0.014424	0.023405	0.044820
		29.673	49.041	53.327	56.341	58.574	59.465
200	3026	0.003791	0.007019	0.011551	0.024244	0.046029	0.098685
		0.004562	0.010350	0.019353	0.040750	0.049974	0.122802
		4.036	7.120	8.502	9.443	10.435	10.675
201	8970	0.000940	0.002395	0.006269	0.011389	0.016991	0.024085
		0.001128	0.002959	0.005412	0.011096	0.017711	0.027170
		18.136	27.520	30.604	32.657	34.039	34.710
202	8978	0.000819	0.003681	0.009003	0.006794	0.014201	0.027807
		0.001409	0.002689	0.006000	0.009792	0.021107	0.037737
		16.864	25.907	29.042	30.764	32.306	33.158
203	7808	0.001036	0.002670	0.005538	0.009814	0.021726	0.031296
		0.001239	0.002861	0.005229	0.008115	0.020425	0.047523
		13.710	21.571	24.405	25.927	27.540	28.491
204	8810	0.000743	0.002662	0.004436	0.009161	0.013179	0.018473
		0.001376	0.002723	0.006247	0.007513	0.015079	0.026746
		15.352	24.025	26.859	28.631	30.083	30.704

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

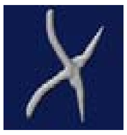
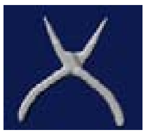
Model	Faces	50%	20%	10%	5%	2%	1%
205	8952	0.000754	0.010514	0.010514	0.023997	0.041278	0.042360
		0.001048	0.002207	0.003992	0.010015	0.022550	0.049710
		15.512	24.535	27.660	29.402	31.055	31.906
206	9584	0.001273	0.002598	0.003547	0.007844	0.014179	0.028297
		0.001459	0.003138	0.004351	0.007814	0.015008	0.032881
		17.225	27.149	31.275	32.497	33.859	35.020
207	10400	0.035034	0.038447	0.032012	0.039465	0.022556	0.029904
		0.001357	0.002428	0.004040	0.007244	0.018602	0.023938
		19.648	31.625	34.269	35.711	37.294	38.395
208	12204	0.004004	0.005200	0.005642	0.007713	0.016251	0.020715
		0.001157	0.002067	0.003196	0.007187	0.014722	0.035440
		22.893	36.773	39.937	41.940	43.543	44.404
209	10216	0.000807	0.002728	0.005080	0.007376	0.019880	0.024560
		0.001093	0.002127	0.004973	0.006494	0.023033	0.027471
		18.196	29.913	32.517	33.919	35.501	36.482
210	11012	0.005697	0.005585	0.011254	0.014997	0.014241	0.019289
		0.001131	0.002359	0.003716	0.007299	0.016559	0.050228
		20.479	33.258	36.062	37.644	39.146	39.987
211	8564	0.000866	0.002945	0.005151	0.027639	0.042656	0.028441
		0.001424	0.002741	0.006083	0.009328	0.021282	0.028368
		17.797	27.984	31.156	32.891	34.031	34.641
212	8592	0.002174	0.005380	0.007314	0.018166	0.015401	0.028371
		0.001409	0.002228	0.004518	0.008396	0.020948	0.028627
		15.963	24.415	27.249	29.132	30.514	31.275

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

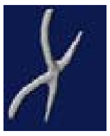
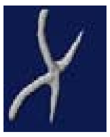
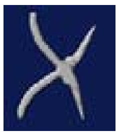
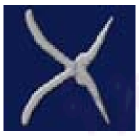
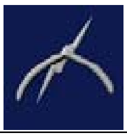
Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
213	7922	0.001063	0.002896	0.004690	0.007707	0.028110	0.035439
		0.001316	0.002499	0.005920	0.008192	0.018778	0.058142
		13.800	21.581	24.315	26.268	27.429	28.421
214	7604	0.001082	0.002811	0.004810	0.010877	0.024160	0.024818
		0.001342	0.002524	0.005242	0.008030	0.018520	0.039676
		13.049	20.560	23.183	24.766	25.847	26.839
215	8922	0.002466	0.015086	0.007353	0.014067	0.015310	0.027283
		0.001130	0.002462	0.006329	0.007721	0.014392	0.028347
		16.634	25.256	28.171	30.053	31.455	32.236
216	8952	0.000941	0.002714	0.004074	0.010171	0.025717	0.033364
		0.001179	0.003289	0.004274	0.008224	0.020619	0.038002
		16.484	25.437	28.801	30.624	32.246	33.168
217	10398	0.002401	0.003685	0.006562	0.006741	0.020918	0.021990
		0.001269	0.002580	0.005086	0.008650	0.013682	0.028925
		19.007	30.934	33.678	35.261	36.593	37.794
218	12392	0.023463	0.032611	0.032614	0.027178	0.033305	0.038607
		0.001092	0.002083	0.004136	0.007145	0.016177	0.026257
		22.873	37.304	40.588	42.681	44.284	45.415
219	7880	0.000827	0.008216	0.009828	0.022546	0.028412	0.038186
		0.001137	0.002473	0.005620	0.009999	0.017513	0.052067
		14.070	21.942	24.806	26.368	27.720	28.731
220	8910	0.001061	0.008559	0.010049	0.016031	0.016101	0.021522
		0.001108	0.001971	0.004370	0.008252	0.019771	0.089288
		16.454	25.407	28.521	30.213	31.736	32.497

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

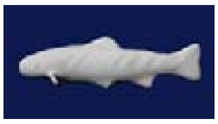
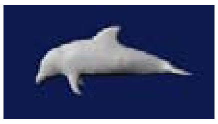
Model	Faces	50%	20%	10%	5%	2%	1%
221	14238	0.000648	0.001990	0.003123	0.007984	0.012909	0.017831
		0.000961	0.002068	0.004672	0.008029	0.023256	0.028294
		22.873	39.437	43.713	46.186	47.468	48.129
222	10506	0.001411	0.009320	0.009320	0.009320	0.014978	0.024195
		0.001160	0.002456	0.004354	0.011620	0.022721	0.044090
		16.714	26.899	31.195	32.467	33.909	34.770
223	13308	0.000597	0.004310	0.004306	0.006909	0.019178	0.019123
		0.001236	0.002236	0.003812	0.011345	0.020348	0.029352
		22.082	38.005	41.730	44.033	45.285	45.916
224	10118	0.000742	0.002394	0.004370	0.008536	0.032093	0.032273
		0.001074	0.002417	0.004574	0.009945	0.018658	0.033964
		16.133	26.158	30.584	32.046	33.598	34.430
225	12148	0.000547	0.001913	0.003997	0.006279	0.016492	0.023795
		0.000846	0.002137	0.003800	0.007389	0.016601	0.027402
		20.520	34.720	38.095	40.158	41.440	42.241
226	10428	0.002137	0.007309	0.007309	0.018669	0.018669	0.025181
		0.001017	0.001978	0.004171	0.011386	0.022534	0.051669
		16.614	26.368	30.604	32.016	33.508	34.379
227	10428	0.002137	0.007309	0.007309	0.018669	0.018669	0.025181
		0.001017	0.001978	0.004171	0.011386	0.022534	0.051669
		16.934	26.708	30.934	32.306	33.779	34.670
228	11180	0.000722	0.002513	0.004323	0.007839	0.019115	0.036614
		0.001078	0.002374	0.004826	0.009839	0.018923	0.042357
		18.677	31.716	34.700	36.312	37.985	38.726

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

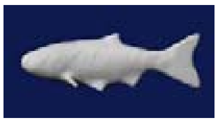
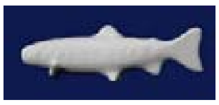
Model	Faces	50%	20%	10%	5%	2%	1%
229	10486	0.000709	0.002412	0.006570	0.009466	0.018026	0.027587
		0.001112	0.002219	0.004104	0.009598	0.025277	0.046216
		17.275	27.119	31.465	32.767	34.239	34.770
230	15142	0.000538	0.002036	0.004006	0.005770	0.014089	0.024276
		0.000881	0.001922	0.003417	0.005962	0.015918	0.031344
		25.186	42.912	46.867	49.401	51.434	52.485
231	12524	0.001431	0.002163	0.004592	0.007849	0.025069	0.024896
		0.000964	0.002802	0.004548	0.007698	0.021810	0.040588
		24.688	42.156	46.625	49.109	50.391	51.109
232	10876	0.000733	0.002287	0.004059	0.009478	0.013723	0.039763
		0.000962	0.002271	0.005051	0.010175	0.025700	0.050585
		18.086	29.993	35.511	37.173	38.866	39.597
233	20368	0.000499	0.001712	0.003636	0.005403	0.013629	0.017744
		0.000904	0.001752	0.003787	0.008843	0.019670	0.034457
		39.734	67.781	76.438	79.719	82.141	82.891
234	14992	0.000739	0.002257	0.004066	0.006887	0.030266	0.032285
		0.000811	0.001844	0.004208	0.009492	0.023266	0.031211
		26.959	46.807	52.215	55.420	57.983	59.035
235	10496	0.000925	0.002508	0.004656	0.008567	0.016183	0.026879
		0.001072	0.002552	0.004339	0.011065	0.027535	0.033380
		19.368	31.906	36.783	38.275	39.927	40.869
236	9686	0.000851	0.002704	0.003929	0.009051	0.013594	0.031620
		0.001134	0.002160	0.004930	0.012126	0.020885	0.038340
		16.604	26.725	31.295	32.627	34.089	34.850

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

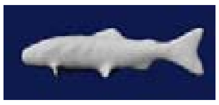
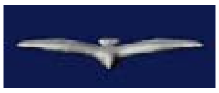
Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
237	8874	0.001707	0.002559	0.004392	0.009588	0.014812	0.040240
		0.000949	0.002839	0.004397	0.010293	0.022345	0.059796
		15.022	26.428	27.099	28.942	30.514	31.245
238	10188	0.000747	0.002250	0.004586	0.009224	0.017315	0.024886
		0.001224	0.002492	0.004824	0.010541	0.024103	0.055552
		17.776	28.882	33.669	35.731	37.534	38.506
239	10238	0.000637	0.002103	0.004139	0.006805	0.014338	0.030576
		0.000947	0.002729	0.004532	0.009056	0.015991	0.029510
		20.156	31.672	36.594	38.031	39.391	40.203
240	13662	0.000656	0.002204	0.003803	0.007423	0.014277	0.039351
		0.001032	0.001913	0.005020	0.009786	0.017315	0.039281
		24.195	40.989	45.065	47.639	49.371	50.122
241	6952	0.000741	0.002159	0.003815	0.007700	0.011164	0.032201
		0.000936	0.002048	0.005365	0.010416	0.021014	0.030753
		14.844	22.578	24.984	26.141	27.328	27.891
242	15694	0.000669	0.002010	0.003406	0.005459	0.011191	0.015059
		0.001111	0.002335	0.003399	0.011136	0.012704	0.022526
		31.578	54.156	59.438	62.766	64.703	65.406
243	6312	0.000846	0.002974	0.005836	0.008503	0.019891	0.023469
		0.001559	0.002748	0.005973	0.012124	0.030985	0.057274
		12.798	20.179	22.693	24.035	25.296	25.827
244	8998	0.000651	0.002418	0.003769	0.007905	0.019052	0.024395
		0.001023	0.002147	0.004078	0.007459	0.022391	0.030340
		18.186	28.291	32.547	33.919	35.231	36.172

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

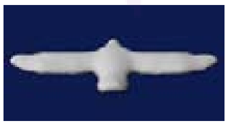
Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
245	12946	0.000876	0.002485	0.003491	0.006766	0.013726	0.023762
		0.001021	0.003197	0.004520	0.007665	0.025258	0.029816
		23.944	41.720	46.016	48.520	49.872	50.593
246	8434	0.000668	0.002759	0.004581	0.006242	0.028944	0.044472
		0.001068	0.003865	0.005899	0.008782	0.018766	0.046088
		16.703	26.922	31.188	32.313	33.641	34.203
247	11034	0.000552	0.001949	0.004194	0.006808	0.010983	0.026658
		0.000819	0.002178	0.003802	0.006673	0.017006	0.027875
		21.912	36.943	40.168	42.401	43.773	44.364
248	23576	0.000624	0.001740	0.002647	0.004012	0.010305	0.020040
		0.000708	0.001165	0.002601	0.005853	0.020244	0.021263
		45.172	77.672	87.562	91.422	94.125	94.984
249	7424	0.000705	0.002456	0.004066	0.008036	0.014107	0.026090
		0.001083	0.002713	0.007054	0.010093	0.020014	0.104541
		15.469	24.375	27.516	29.125	29.953	30.484
250	8556	0.002148	0.009327	0.009327	0.012026	0.015401	0.015401
		0.000793	0.002214	0.007486	0.011923	0.027166	**0.313558**
		17.906	28.110	31.525	33.348	34.730	35.781
251	6312	0.000846	0.002974	0.005836	0.008503	0.019891	0.023469
		0.001559	0.002748	0.005973	0.012124	0.030985	0.057274
		12.829	21.391	23.813	24.969	25.922	26.297
252	12698	0.000575	0.001985	0.003797	0.005662	0.014313	0.017741
		0.001013	0.002340	0.003740	0.005961	0.013490	0.032299
		24.391	41.922	45.797	48.234	49.625	50.438

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

Model	Faces	50%	20%	10%	5%	2%	1%
253	12324	0.001322	0.001908	0.003488	0.005515	0.014874	0.021809
		0.000788	0.002258	0.004739	0.007763	0.021712	0.029519
		19.391	32.593	35.453	36.984	37.719	37.984
254	7272	0.000647	0.002251	0.003952	0.006875	0.024473	0.017940
		0.000969	0.002425	0.004690	0.011995	0.038387	0.040786
		12.766	21.031	23.453	24.797	25.375	25.688
255	10104	0.000722	0.002241	0.003374	0.007398	0.011686	0.020710
		0.001161	0.001972	0.004945	0.009833	0.017489	0.034060
		18.781	29.734	34.484	35.656	36.875	37.391
256	11936	0.011743	0.016178	0.018178	0.018178	0.018178	0.028971
		0.000731	0.001591	0.004017	0.011878	0.024481	0.106918
		23.203	39.203	42.734	44.562	45.453	46.000
257	4266	0.036795	0.051530	0.051530	0.076139	0.094254	**0.192701**
		0.005563	0.015646	0.079521	**0.091119**	0.109779	0.109779
		8.272	11.937	13.059	13.790	14.781	15.342
258	17888	0.004323	0.005595	0.008951	0.011477	0.019433	0.031010
		0.002644	0.012232	0.028545	**0.064749**	0.068432	0.095392
		33.672	57.531	62.656	65.563	66.547	67.172
259	18026	0.000669	0.001877	0.003234	0.005356	0.014701	0.026121
		0.000831	0.001899	0.003629	0.005445	0.014436	0.023062
		35.361	57.713	62.760	66.355	68.368	69.470

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont’d)

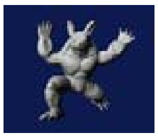
Model	Faces	50%	20%	10%	5%	2%	1%
260	4990	0.003781	0.012264	0.012911	0.042963	0.092943	**0.247300**
		0.003521	0.009942	0.025309	0.043397	0.067166	**0.426666**
		9.223	13.089	14.741	15.542	16.113	16.343
281	50542	0.002348	0.004108	0.004852	0.006469	0.018270	0.031355
		0.001501	0.002587	0.004435	0.006124	0.020837	0.035884
		105.682	178.196	197.905	207.599	211.694	213.697
282	48942	0.002499	0.004850	0.006796	0.008690	0.011651	0.029683
		0.001997	0.002564	0.005978	0.007660	0.024542	0.040134
		95.748	166.399	187.149	196.753	201.580	203.793
283	48964	0.002163	0.002746	0.003583	0.005194	0.015780	0.025718
		0.001257	0.002137	0.003194	0.009889	0.023408	0.043920
		99.797	166.625	186.203	196.109	200.313	202.234
284	50984	0.002366	0.002584	0.004040	0.005814	0.010392	0.032316
		0.001365	0.002342	0.006288	0.012321	0.056800	0.109229
		92.393	157.086	181.611	193.528	216.902	222.760
285	50382	0.002438	0.003416	0.003906	0.006883	0.011586	0.023389
		0.002339	0.002293	0.004362	0.007357	0.020292	0.037635
		101.406	169.063	191.355	202.272	208.129	210.042
286	50212	0.002646	0.004118	0.006273	0.007761	0.014231	0.032564
		0.001272	0.002750	0.003917	0.007078	0.016418	0.036690
		109.708	193.569	217.493	230.241	235.298	237.752
287	50978	0.002910	0.004797	0.004494	0.006312	0.014605	0.027620
		0.001446	0.002862	0.006601	0.016523	0.055717	0.102019
		101.188	177.188	197.391	209.156	214.109	214.625

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

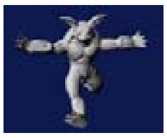
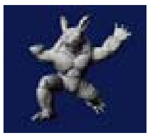
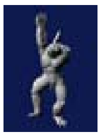
Model	Faces	50%	20%	10%	5%	2%	1%
288	39996	0.003453	0.003914	0.004418	0.006517	0.020426	0.031706
		0.001624	0.003178	0.004879	0.009535	0.025422	0.040580
		72.004	119.922	134.413	141.924	145.229	146.691
289	50418	0.001276	0.002816	0.004181	0.006197	0.009815	0.024683
		0.001162	0.002310	0.003344	0.005411	0.016902	0.038586
		91.031	152.079	169.804	178.907	182.803	184.736
290	50858	0.002652	0.004500	0.004642	0.006121	0.012543	0.026452
		0.001417	0.002939	0.005999	0.021006	0.075459	0.106535
		88.768	147.272	164.547	173.700	177.555	179.128
291	49226	0.003680	0.006228	0.008930	0.008930	0.013989	0.030029
		0.001786	0.003055	0.004644	0.007911	0.019760	0.037161
		86.855	144.988	162.183	171.186	174.421	176.033
292	43544	0.002236	0.003880	0.004922	0.007131	0.018535	0.032799
		0.001522	0.002625	0.004391	0.009743	0.019232	0.042093
		76.891	128.995	143.997	149.425	154.392	155.714
293	29996	0.003477	0.005483	0.005498	0.008854	0.022563	0.033604
		0.001867	0.003809	0.005902	0.010018	0.023146	0.059177
		53.307	88.928	99.994	104.160	107.264	108.506
294	32998	0.001409	0.003387	0.003852	0.006517	0.015962	0.031869
		0.001933	0.002879	0.006016	0.009254	0.024699	0.041768
		56.101	94.125	105.271	109.438	112.802	114.174
295	33556	0.003288	0.003572	0.004888	0.007241	0.021011	0.040401
		0.001924	0.002685	0.004783	0.008635	0.025635	0.049961
		56.782	88.978	102.878	109.988	113.213	114.795

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

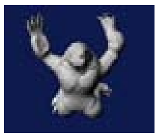
Model	Faces	50%	20%	10%	5%	2%	1%
296	31660	0.001483	0.003964	0.005515	0.008954	0.012913	0.041018
		0.001830	0.003246	0.005380	0.009926	0.023292	0.038529
		53.657	83.420	96.619	103.289	106.163	107.204
297	30626	0.002259	0.003476	0.005544	0.008919	0.016042	0.044522
		0.001920	0.003213	0.005722	0.012578	0.025501	0.043657
		51.644	86.745	96.935	100.394	103.078	104.140
298	29996	0.003320	0.003892	0.005609	0.009479	0.023094	0.034523
		0.001802	0.003373	0.006062	0.010766	0.025550	0.044063
		52.095	87.075	97.580	101.506	104.430	105.532
299	34230	0.001299	0.003399	0.004930	0.006905	0.013447	0.039935
		0.001575	0.003144	0.005006	0.010636	0.029246	0.062591
		57.963	97.190	108.726	112.872	116.137	117.489
300	34902	0.001034	0.002850	0.004125	0.006575	0.013718	0.034051
		0.001187	0.002840	0.003849	0.008825	0.020463	0.035228
		64.473	115.636	124.769	136.076	145.940	150.076
301	18500	0.001279	0.003543	0.005097	0.008633	0.012728	0.030864
		0.001613	0.003367	0.004583	0.007732	0.021307	0.045735
		33.328	56.371	61.378	64.473	66.235	66.866
302	50246	0.008379	0.008379	0.009757	0.009757	0.011909	0.018582
		0.004788	0.004853	0.004627	0.009664	0.035713	0.086761
		103.489	170.806	190.915	201.870	208.710	209.872
303	31028	0.006967	0.008942	0.011577	0.007261	0.013702	0.025727
		0.001456	0.002683	0.005509	0.010627	0.021562	0.038396
		56.321	85.243	97.570	103.509	105.502	106.123

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

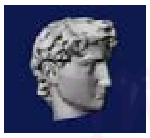
Model	Faces	50%	20%	10%	5%	2%	1%
304	50930	0.000599	0.001842	0.002798	0.007719	0.011441	0.018507
		0.001140	0.001873	0.003466	0.005419	0.015761	0.020099
		82.348	143.657	162.053	171.356	174.841	176.324
305	55644	0.000654	0.004173	0.006765	0.007604	0.007487	0.010833
		0.000963	0.001926	0.002762	0.005794	0.011978	0.029618
		98.021	162.814	182.042	192.016	197.103	197.945
306	53592	0.000548	0.001222	0.004967	0.005063	0.008084	*0.010511*
		0.000783	0.001474	0.002459	0.002999	0.009116	*0.012589*
		95.437	158.228	176.384	185.587	189.122	190.774
307	54874	0.000553	0.001255	0.002672	0.005348	*0.007113*	0.013534
		0.000919	0.001529	0.002337	0.003576	0.008647	*0.012462*
		101.155	165.498	184.305	193.869	197.634	199.307
308	47948	0.000660	0.001478	0.002528	0.005099	0.009404	0.013855
		0.000846	0.001543	0.002261	0.003499	0.008740	0.014985
		82.819	137.718	153.300	161.372	164.376	165.558
309	10390	0.005295	0.011091	0.016269	0.030939	0.070816	0.116687
		**0.007843**	**0.020992**	0.037047	0.090530	0.091532	0.138141
		17.866	28.541	31.676	32.997	34.190	34.600
310	16522	0.001870	0.003964	0.006398	0.015326	0.017712	0.031271
		0.002066	0.003257	0.005543	0.010715	0.020151	0.037356
		28.841	43.933	51.614	53.217	55.069	55.440
311	50456	0.000573	0.002063	0.007105	0.007685	0.016879	0.020937
		0.001211	0.002245	0.003470	0.005399	0.010804	0.020043
		100.104	171.947	193.108	203.703	208.300	209.942

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

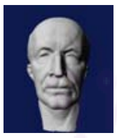
Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
312	53966	0.003791	0.004310	0.004290	0.006515	0.013657	0.021130
		0.000868	0.001824	0.003319	0.005230	0.016325	0.031007
		100.655	182.412	208.209	222.520	230.622	234.197
313	21700	0.003873	0.013893	0.013893	0.013893	0.033479	0.049448
		0.002407	0.004258	0.007805	0.012166	0.026915	0.048070
		48.940	79.514	85.503	89.238	91.602	92.173
314	52870	0.000590	0.001564	0.003118	0.004457	0.008056	0.012410
		0.000994	0.001565	0.002488	0.004626	0.010092	0.023604
		103.399	173.760	192.271	206.136	211.875	213.177
315	51532	0.003816	0.003821	0.003715	0.005315	0.009247	0.015443
		0.000969	0.001637	0.003345	0.004913	0.010060	0.018154
		88.467	158.118	178.296	188.261	195.538	194.530
316	46786	0.000822	0.002284	0.007550	0.007779	0.011603	0.016457
		0.001450	0.002298	0.004297	0.007502	0.015187	0.023404
		84.061	150.466	168.763	177.886	182.012	184.015
317	55448	0.000411	0.001381	0.003593	0.006221	0.010216	0.013079
		0.000751	0.001434	0.002124	0.004094	0.007899	0.016138
		100.895	170.525	191.315	202.752	208.340	209.451
318	50286	0.000837	0.001865	0.002827	0.004289	0.009824	0.018536
		0.000824	0.001393	0.002367	0.005788	0.010495	0.019303
		99.433	167.431	187.890	198.255	204.334	208.259
319	53112	0.000293	0.000931	*0.001607*	0.005928	0.008381	*0.009600*
		0.000776	0.000847	0.001439	0.003014	*0.006337*	*0.009632*
		111.563	188.547	210.125	220.719	224.609	226.453

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

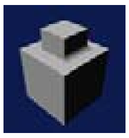
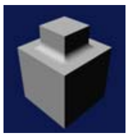
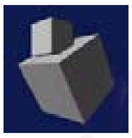
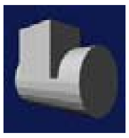
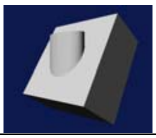
Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
320	54232	0.000477	0.002135	0.004257	0.004663	*0.007022*	0.011573
		0.001069	0.001587	0.002688	0.004089	0.010056	0.017811
		115.734	196.938	220.641	232.234	236.625	238.578
321	29986	0.015154	0.080697	0.080336	0.082307	0.082629	0.085642
		*0.000000*	*0.000000*	*0.000005*	*0.000006*	0.025466	0.120618
		81.838	123.698	134.443	138.860	141.503	142.204
322	29978	0.000110	**0.111473**	0.111774	0.111540	0.114065	0.113415
		0.000000	*0.000000*	*0.000005*	*0.000013*	0.025211	0.112454
		77.281	120.000	132.50	137.125	140.250	140.875
323	29978	*0.000005*	0.002503	**0.126121**	**0.126160**	**0.126272**	0.125445
		*0.000000*	*0.000000*	*0.000006*	0.000086	0.012144	0.096118
		72.845	109.668	120.623	125.030	128.124	128.855
324	29620	**0.087634**	0.088982	0.088435	0.088198	0.088184	0.088278
		0.000008	0.000776	0.002046	0.006336	0.023576	0.055453
		53.016	85.162	95.567	99.473	101.726	102.237
325	29994	0.000418	0.000695	*0.000658*	*0.000695*	*0.001428*	*0.001428*
		0.000005	0.000006	0.000008	0.000064	0.023785	0.088748
		55.099	91.682	103.539	110.028	112.352	112.862
326	29986	0.057047	0.057047	0.057059	0.057059	0.057008	0.057008
		0.000005	0.000015	0.000025	0.000807	0.058944	0.106564
		57.152	93.735	104.851	110.769	112.271	113.133
327	29870	0.000032	0.000076	*0.000205*	*0.000753*	*0.000951*	*0.004480*
		0.000006	0.000023	0.000263	0.016248	0.176316	0.173976
		47.919	81.527	92.393	97.871	99.103	100.024

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

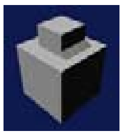
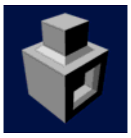
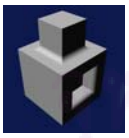
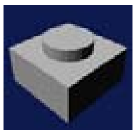
Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
328	29984	*0.000005*	0.000006	**0.128393**	**0.128678**	**0.127983**	0.128620
		0.000000	*0.000000*	*0.000005*	*0.000007*	0.028707	0.082320
		79.264	123.398	136.576	141.804	145.159	145.910
329	29994	0.016464	**0.124097**	**0.123632**	**0.123632**	0.124106	0.124066
		0.000000	*0.000000*	*0.000000*	*0.000005*	*0.000007*	0.023464
		79.594	120.984	132.190	136.877	139.941	140.552
330	29996	*0.000006*	**0.095017**	0.094942	0.094942	0.094707	0.094762
		*0.000000*	*0.000000*	*0.000005*	*0.000007*	0.030605	0.092813
		75.047	120.406	132.766	137.234	139.766	140.281
331	29996	**0.113543**	**0.113543**	**0.113543**	0.113543	0.113543	0.113543
		*0.000000*	*0.000000*	*0.000005*	*0.000007*	0.020965	0.076975
		77.031	128.938	143.719	148.719	151.781	153.203
332	29990	0.000007	0.066778	0.065047	0.074340	0.095718	0.117880
		0.000000	0.000007	0.000020	*0.000052*	0.027269	0.092432
		73.025	100.434	112.752	119.362	122.336	123.067
333	29880	**0.113542**	**0.113542**	0.113542	0.113542	0.113542	0.113542
		*0.000000*	0.000006	0.000011	0.000310	0.016547	0.111865
		66.776	96.419	107.505	111.300	114.084	114.855
334	29990	**0.088971**	0.088971	0.088971	0.088971	0.088971	0.088971
		*0.000000*	*0.000000*	*0.000006*	0.000320	0.086556	0.127409
		65.969	106.953	121.016	128.281	129.484	130.375
335	29986	*0.000006*	0.087710	**0.122494**	**0.122601**	0.122500	0.121908
		0.000000	*0.000000*	*0.000004*	*0.000006*	0.022227	0.139162
		75.281	119.141	131.125	135.625	138.422	138.891

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

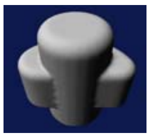
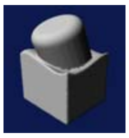
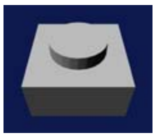
Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
336	31006	0.004041	0.045860	0.045616	0.045805	0.045674	0.045440
		0.000000	0.001304	0.001940	0.002615	*0.004845*	*0.010092*
		59.896	92.583	104.460	110.679	112.902	113.663
337	23062	0.000478	0.001428	0.002649	0.005906	0.011273	0.023641
		0.000877	0.002302	0.002716	0.004794	0.011297	0.029389
		40.779	69.610	78.413	81.728	83.790	84.361
338	38802	0.000116	0.007732	0.006889	0.006769	*0.006752*	*0.009646*
		0.000055	0.001297	0.001858	0.003714	0.014642	0.017717
		70.141	116.357	132.140	140.472	143.967	145.379
339	29996	**0.113543**	**0.113543**	**0.113543**	**0.113543**	0.113543	0.113543
		0.000000	0.000001	*0.000005*	*0.000007*	*0.000007*	0.104618
		63.922	106.250	119.093	123.672	126.172	126.688
340	29990	**0.088965**	0.088965	0.088965	0.088965	0.088969	0.088970
		*0.000000*	*0.000000*	*0.000006*	*0.000006*	0.081418	0.106384
		64.234	104.844	118.250	125.578	126.781	127.688
341	3322	*0.000001*	*0.000001*	0.009260	0.020886	0.038773	0.078833
		0.025949	**0.115654**	**0.195246**	**0.197037**	**0.197037**	0.197037
		4.777	7.751	8.512	8.943	9.303	9.544
342	3322	*0.000000*	*0.000000*	0.006272	0.015127	0.038036	0.065457
		0.032470	**0.184711**	**0.186176**	**0.190001**	**0.211895**	0.215702
		4.578	7.313	7.844	8.093	8.250	8.343
343	3322	*0.000001*	*0.000001*	0.006272	0.017890	0.038036	0.076702
		0.019264	**0.135576**	**0.135576**	**0.141781**	0.142812	0.142812
		4.356	6.699	7.150	7.431	7.611	7.711

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

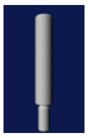
Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
344	6360	0.002923	0.013797	0.013797	0.016082	0.029316	0.044245
		0.001634	0.003064	0.006492	0.023473	0.057499	0.108977
		11.436	17.045	18.457	19.798	20.770	21.210
345	7224	0.007032	0.010248	0.011395	0.017930	0.035908	0.040513
		0.001160	0.003843	0.008895	0.030783	**0.187655**	0.225259
		13.239	19.919	22.062	23.183	23.904	24.065
346	7170	0.003173	0.003634	0.009484	0.014055	0.027988	0.042094
		0.002211	0.002766	0.005800	0.020012	0.085879	0.189056
		13.159	19.839	21.932	23.053	24.075	24.295
347	5242	0.002552	0.005206	0.009919	0.030132	0.073209	0.113882
		0.001710	0.005011	0.009622	0.048793	0.088630	0.152767
		9.664	13.490	15.542	16.594	17.215	17.375
348	5132	*0.000001*	*0.000001*	*0.000001*	*0.002773*	0.012190	0.035879
		0.073313	**0.082654**	**0.082654**	**0.220317**	**0.220317**	0.235518
		6.189	11.847	12.498	13.319	13.740	13.900
349	5132	*0.000000*	*0.000001*	*0.000001*	*0.002956*	0.017762	0.026556
		0.071576	**0.216260**	**0.220180**	**0.220180**	**0.223769**	0.231107
		6.269	11.537	12.328	13.089	13.449	13.600
350	17514	0.021003	0.021692	0.021695	0.028357	0.035002	0.035871
		0.000231	0.001401	0.001951	0.004810	0.019150	0.074257
		30.284	45.085	52.175	54.408	55.470	56.121
351	10380	**0.222535**	**0.458087**	**0.593273**	**0.796202**	**0.803227**	**0.866060**
		0.006076	0.001245	**0.084882**	N/A*	N/A*	N/A*
		18.987	35.030	36.693	N/A*	N/A*	N/A*

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

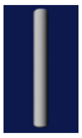
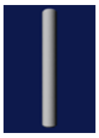
Model	Faces	50%	20%	10%	5%	2%	1%
352	20796	0.057031	0.057605	0.057547	0.056908	0.057274	0.057824
		0.000420	0.000812	0.001399	0.003421	0.008715	0.023677
		37.524	56.141	64.643	67.818	70.001	70.822
353	3020	**0.093447**	**0.093447**	**0.353277**	**0.451729**	**0.461662**	**0.461662**
		0.066951	**0.287812**	**0.287812**	**0.287812**	**0.287812**	**0.287812**
		4.066	6.099	6.610	6.900	7.080	7.180
354	5816	0.004914	0.019172	0.019172	0.026197	0.026197	0.042667
		0.002933	0.003967	0.009031	0.049494	0.056885	0.108334
		9.794	15.172	16.634	18.097	19.007	19.188
355	29984	0.000007	0.000236	*0.000246*	*0.000236*	*0.000246*	*0.001971*
		0.000009	0.000086	0.000086	0.000689	*0.000245*	*0.005283*
		47.900	71.052	81.437	84.822	86.865	87.776
356	29980	0.000007	*0.000010*	*0.000032*	*0.000183*	*0.000183*	*0.001971*
		0.000010	0.000052	0.000183	0.000183	*0.000183*	*0.007412*
		48.009	71.663	82.358	85.813	87.866	88.808
357	29764	0.050152	0.049930	0.049664	0.049186	0.049365	0.050016
		0.000000	0.000083	0.000725	0.004558	0.022579	0.056077
		59.656	88.657	100.404	106.693	108.596	109.057
358	28266	**0.492324**	**0.492324**	**0.492324**	**0.584723**	**0.584723**	**0.617882**
		0.000006	0.000007	0.000495	0.007789	0.059920	0.068989
		50.813	82.769	89.719	95.607	97.540	98.001
359	29924	0.059355	0.059988	0.059359	0.059987	0.059988	0.059987
		0.000005	0.000008	0.000241	0.013123	0.059987	0.186906
		54.498	87.866	97.990	101.646	104.060	104.701

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

Model	Faces	50%	20%	10%	5%	2%	1%
360	4396	0.007333	0.007333	0.014286	0.029113	0.054777	0.127590
		0.002018	0.005585	0.014512	0.044591	0.070872	0.079955
		5.318	9.944	11.216	12.258	12.738	12.869
361	29734	0.001089	0.002497	0.005113	0.006773	0.017582	0.025099
		0.001337	0.002258	0.005189	0.011103	0.038002	0.081720
		49.631	88.770	93.965	97.891	100.955	102.267
362	28952	0.001016	0.002465	0.004162	0.004712	0.014114	0.020948
		0.001324	0.002366	0.003053	0.006038	0.014147	0.027846
		48.360	81.657	91.361	94.836	97.360	98.031
363	26864	0.000512	0.001392	0.002494	0.003539	0.007769	0.011731
		0.000625	0.001052	0.001997	0.002943	*0.006675*	*0.011746*
		42.792	66.736	77.041	82.128	84.211	84.992
364	27100	0.000866	0.001734	0.004407	0.006426	0.013439	0.030261
		0.001087	0.002190	0.003384	0.007570	0.013653	0.046054
		43.493	73.566	82.138	85.303	87.326	87.926
365	27028	0.001199	0.001830	0.003469	0.005283	0.009958	0.017085
		0.001314	0.001381	0.002479	0.005530	0.015108	0.031040
		44.534	74.447	82.999	85.964	87.997	88.758
366	21274	0.000751	0.002113	0.003610	0.008144	0.017943	0.027167
		0.001263	0.001994	0.003158	0.006916	0.016798	0.034307
		35.471	59.696	66.846	69.840	71.673	72.284
367	7800	0.001031	0.003679	0.008725	0.013665	0.023664	0.051209
		0.001760	0.003834	0.008781	0.020098	0.066590	0.101866
		12.638	22.062	23.384	24.946	26.037	26.618

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
368	22400	0.001136	0.001877	0.003279	0.005355	0.017525	0.028784
		0.001313	0.001834	0.003335	0.007202	0.017397	0.031721
		36.563	57.022	65.694	68.949	71.082	71.763
369	27212	0.001073	0.002380	0.003364	0.007818	0.012758	0.026730
		0.001365	0.002311	0.003517	0.007333	0.026608	0.040568
		44.284	68.699	79.895	85.233	87.265	88.067
370	27836	0.000654	0.001730	0.002450	0.005440	0.009909	0.017217
		0.000810	0.001302	0.002695	0.004708	0.012370	0.026858
		45.726	76.190	82.198	88.087	90.230	91.141
371	29194	0.000720	0.000967	0.001958	0.003532	*0.007044*	0.014705
		0.000681	0.001358	0.002228	0.003419	0.007498	0.017142
		58.394	87.265	98.652	104.530	107.044	108.426
372	13368	0.000711	0.002705	0.004522	0.007663	0.022497	0.031214
		0.001097	0.001982	0.004720	0.009001	0.027446	0.052002
		20.219	35.040	38.405	40.358	41.570	42.401
373	21112	0.000677	0.001791	0.003523	0.005444	0.015201	0.036518
		0.001003	0.001588	0.003104	0.005905	0.015363	0.038300
		33.568	57.843	65.084	67.848	69.830	70.501
374	29744	0.000836	0.001552	0.002914	0.005650	0.009434	0.016347
		0.000847	0.001663	0.003087	0.004776	0.010111	0.021371
		47.408	80.135	86.815	92.713	95.047	95.778
375	23520	0.000900	0.002230	0.003098	0.004981	0.018029	0.021959
		0.001342	0.002366	0.003528	0.007525	0.022119	0.034241
		38.465	64.703	72.174	74.747	76.230	77.111

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont’d)

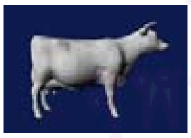
Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
376	25132	0.000944	0.002388	0.002732	0.004151	0.008665	0.016223
		0.001030	0.002410	0.003047	0.006917	0.018158	0.023121
		41.640	70.071	78.283	80.967	82.669	83.160
377	30342	0.001096	0.001808	0.003805	0.007788	0.012875	0.021184
		0.000940	0.002056	0.003961	0.008046	0.022772	0.044124
		47.999	81.607	91.401	95.007	97.510	98.331
378	13368	0.000676	0.002950	0.004083	0.006771	0.020130	0.026535
		0.001477	0.002387	0.004373	0.009897	0.021195	0.055825
		20.309	35.100	38.105	40.018	41.179	41.970
379	13842	0.000663	0.002632	0.005081	0.009778	0.022656	0.037411
		0.000975	0.002020	0.003856	0.009788	0.022203	0.050516
		21.952	37.624	41.149	43.312	44.774	45.495
380	18708	0.001488	0.002489	0.004994	0.009228	0.014456	0.045442
		0.001032	0.002652	0.003760	0.010395	0.026336	0.051889
		33.238	56.641	64.172	66.746	68.418	68.939
381	13872	0.001375	0.007602	0.013652	0.016166	0.054305	0.076122
		0.001673	0.005275	0.007882	0.017825	0.055342	0.068584
		28.150	46.206	50.483	52.846	54.398	55.430
382	16152	0.000992	0.002880	0.005779	0.011237	0.021359	0.029678
		0.001302	0.002748	0.005367	0.012228	0.028687	0.077043
		26.128	44.404	48.160	50.563	52.035	52.616
383	26658	0.001858	0.003464	0.004870	0.013298	0.015997	0.043755
		0.001285	0.002411	0.005471	0.012166	0.026659	0.041848
		42.581	66.395	76.961	82.068	84.051	84.632

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

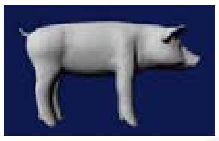
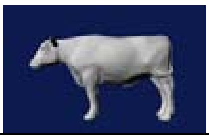
Model	Faces	50%	20%	10%	5%	2%	1%
384	9420	0.001663	0.006538	0.008346	0.014631	0.035855	0.075665
		0.002395	0.005217	0.008186	0.021023	0.031447	0.071258
		14.861	23.814	26.758	28.281	29.282	29.783
385	20868	0.000838	0.002957	0.005476	0.009009	0.029864	0.084266
		0.000972	0.003372	0.005985	0.011942	0.027239	0.044473
		32.497	55.330	62.049	64.333	65.624	66.355
386	4170	0.004953	0.009793	0.016621	0.064145	**0.208613**	**0.208613**
		0.008591	**0.020116**	0.028540	0.081359	**0.212535**	**0.333917**
		4.677	8.873	9.734	10.335	10.866	11.076
387	29356	0.001796	0.002954	0.004167	0.006567	0.018617	0.021281
		0.001247	0.001976	0.004374	0.008457	0.019172	0.035760
		47.538	80.255	89.769	93.154	95.477	96.248
388	16818	0.000983	0.003286	0.009164	0.018063	0.020775	0.052769
		0.001737	0.003315	0.007386	0.014566	0.028490	0.047109
		26.859	45.195	49.331	51.965	53.737	54.258
389	18976	0.001569	0.004104	0.009458	0.016880	0.030284	0.050725
		0.001978	0.004735	0.010005	0.018903	0.034803	0.061026
		30.083	51.144	55.410	58.174	59.806	60.547
390	18474	0.004718	0.004718	0.007415	0.009838	0.022606	0.041720
		0.001415	0.002932	0.004859	0.010848	0.032931	0.074411
		29.613	49.691	53.898	56.712	58.364	59.084
391	8626	0.002365	0.006586	0.017498	0.021369	0.047277	0.082662
		0.003351	0.006442	0.013083	0.031258	0.051090	0.088611
		14.551	22.643	25.266	26.448	27.810	28.571

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

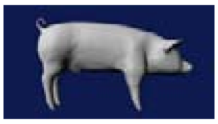
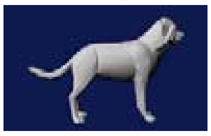
Model	Faces	50%	20%	10%	5%	2%	1%
392	17232	0.001389	0.002923	0.006117	0.009981	0.038493	0.053960
		0.001406	0.002271	0.005318	0.011862	0.034991	0.047618
		27.319	46.647	50.833	53.547	55.390	56.021
393	19510	0.001035	0.007336	0.007337	0.010655	0.021177	0.034376
		0.001885	0.003904	0.009280	0.020976	0.038926	0.083584
		31.996	54.158	58.845	62.720	63.752	64.483
394	18014	0.002032	0.006436	0.006917	0.011420	0.016697	0.040365
		0.001495	0.002989	0.007534	0.013326	0.044649	0.082361
		28.331	44.063	50.843	53.417	54.929	55.410
395	22620	**0.272984**	**0.332769**	**0.332769**	**0.332769**	**0.385494**	**0.398229**
		0.002030	0.007206	0.017601	0.028308	0.147329	0.161227
		36.482	56.671	67.948	70.541	71.533	71.893
396	11072	0.010332	0.010332	0.019134	0.040051	0.040983	0.042829
		0.001887	0.004850	0.010394	0.024808	0.058231	0.071051
		18.477	31.445	33.909	35.190	36.432	37.083
397	14532	0.001225	0.003593	0.007052	0.020791	0.025646	0.073059
		0.001495	0.003482	0.009261	0.018979	0.025230	0.069649
		23.133	39.877	43.352	45.495	46.937	47.528
398	2956	0.004507	0.014878	0.016821	0.027920	**0.131701**	**0.186786**
		**0.009676**	**0.033536**	**0.053987**	**0.101506**	0.129866	**0.249447**
		3.705	6.169	6.800	7.421	8.142	8.332
399	7818	0.002028	0.006164	0.010386	0.016235	0.049335	0.061937
		0.002126	0.005863	0.010436	0.021605	0.042028	0.096367
		13.059	20.390	23.023	24.465	25.136	25.647

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

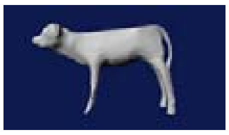
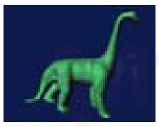
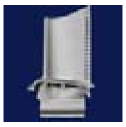
Model	**Faces**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
400	7402	0.002110	0.004891	0.011562	0.018601	0.038060	0.129511
		0.002438	0.006347	0.013551	0.033655	0.091739	0.152347
		12.137	19.188	21.361	22.392	23.524	24.025
Angel	473986	0.000651	0.000840	*0.001590*	*0.002387*	*0.004977*	*0.004795*
		0.000166	0.000530	0.000802	0.001514	*0.003639*	*0.005333*
		1724.938	2765.348	3016.984	3127.703	3195.563	3208.172
Armadillo	345944	0.000308	0.000801	*0.001390*	*0.002438*	*0.004754*	*0.006938*
		0.000395	0.000766	0.001657	0.002386	*0.004894*	*0.007601*
		988.091	1604.988	1753.251	1819.687	1858.823	1874.616
Bunny	10000	0.005666	0.005851	0.011929	0.018848	0.037970	0.063597
		0.005000	0.007320	0.014521	0.024966	0.058851	0.115279
		18.607	28.882	32.807	39.267	40.298	40.889
Canyon	123008	0.002509	0.007535	0.013386	0.013422	0.018751	0.021248
		0.004899	0.012800	0.028362	0.033196	0.033319	0.034957
		252.423	405.323	451.109	473.982	486.449	490.535
Dinosaur	47903	0.000415	0.001216	0.003127	0.006570	0.008190	0.012105
		0.000809	0.001367	0.002209	0.004090	0.010709	0.021204
		83.660	134.714	155.874	170.926	182.913	188.481
Dragon	147572	0.000390	0.001371	0.002517	0.004044	0.008756	0.013321
		0.000924	0.001713	0.002734	0.005545	0.012302	0.023209
		321.532	548.629	615.195	646.279	662.993	669.493
Horse	96966	0.002390	0.002390	0.004419	0.004419	0.008960	0.014587
		0.000490	0.001019	0.004283	0.004283	*0.005694*	0.014777
		192.046	308.654	341.121	357.724	367.058	369.932

Table A-1: Hausdorff distances and running times of models simplified with QEM and our algorithm (cont'd)

Model	Faces	50%	20%	10%	5%	2%	1%
Turbine	239934	0.012420	0.012651	0.023771	0.023771	0.024878	0.025907
		0.001682	0.002096	0.005963	0.016141	0.053420	0.082872
		642.264	1071.190	1228.817	1276.355	1302.863	1315.271
Average	QEM	0.007246	0.011695	0.015607	0.020497	0.030282	0.043810
	Ours	0.002154	0.006647	0.010642	0.017894	0.036530	0.061322
Max	QEM	0.492324	0.492324	0.593273	0.796202	0.803227	0.866060
	Ours	0.073313	0.287812	0.287812	0.287812	0.309078	0.487676
Min	QEM	0.000000	0.000000	0.000001	0.000183	0.000183	0.001428
	Ours	0.000000	0.000000	0.000000	0.000005	0.000007	0.005283

Bold indicates 10 highest Hausdorff distances at the given percentage for the given algorithm

Italics indicate 10 lowest values at the given percentage for the given algorithm

* N/A: The algorithm exhausted all possible contractions before the given level

Table A-2: Penalty weights for all models

Model	α	β	δ	**Model**	α	β	δ	**Model**	α	β	δ	**Model**	α	β	δ
1	4	0	4	45	4	0	1	89	4	0	1	133	4	0	1
2	4	0	1	46	4	0	1	90	4	0	1	134	4	0	1
3	4	0	1	47	4	0	1	91	4	0	1	135	4	0	1
4	4	0	1	48	4	0	1	92	4	0	1	136	4	0	1
5	4	0	1	49	4	0	1	93	4	0	1	137	4	0	1
6	4	0	1	50	4	0	1	94	4	0	1	138	4	0	1
7	4	0	1	51	4	0	1	95	4	0	1	139	4	0	1
8	4	0	1	52	4	0	1	96	4	0	1	140	4	0	1
9	4	0	1	53	4	0	1	97	4	0	1	141	4	0	1
10	4	0	1	54	4	0	1	98	4	0	1	142	4	0	1
11	4	0	2	55	4	0	1	99	4	0	1	143	4	0	1
12	4	0	1	56	4	0	1	100	4	0	1	144	4	0	1
13	4	0	1	57	4	0	1	101	4	0	1	145	4	0	1
14	4	0	1	58	4	0	1	102	4	0	1	146	4	0	1
15	4	0	1	59	4	0	1	103	4	0	1	147	4	0	1
16	4	0	1	60	4	0	1	104	4	0	1	148	4	0	1
17	4	0	1	61	4	0	2	105	4	0	1	149	4	0	1
18	4	0	1	62	4	0	1	106	4	0	1	150	4	0	1
19	4	0	1	63	4	0	1	107	4	0	1	151	4	0	1
20	4	0	1	64	4	0	1	108	4	0	1	152	4	0	1
21	4	0	2	65	4	0	1	109	4	0	1	153	4	0	1
22	4	0	1	66	4	0	1	110	4	0	1	154	4	0	1
23	4	0	1	67	4	0	1	111	4	0	2	155	4	0	1
24	4	0	1	68	4	0	1	112	4	0	1	156	4	0	1
25	4	0	1	69	4	0	1	113	4	0	1	157	4	0	1
26	4	0	1	70	4	0	1	114	4	0	1	158	4	0	1
27	4	0	1	71	4	0	1	115	4	0	1	159	4	0	1
28	4	0	1	72	4	0	1	116	4	0	1	160	4	0	1
29	4	0	1	73	4	0	1	117	4	0	1	161	4	0	1
30	4	0	1	74	4	0	1	118	4	0	1	162	4	0	1
31	6	0	2	75	4	0	1	119	4	0	1	163	4	0	1
32	4	0	1	76	4	0	1	120	4	0	1	164	4	0	1
33	4	0	1	77	4	0	1	121	4	0	1	165	4	0	1
34	4	0	1	78	4	0	1	122	4	0	1	166	4	0	1
35	4	0	1	79	4	0	1	123	4	0	1	167	4	0	1
36	4	0	1	80	4	0	1	124	4	0	1	168	4	0	1
37	4	0	1	81	8	0	1	125	4	0	1	169	4	0	1
38	4	0	1	82	4	0	1	126	4	0	1	170	4	0	1
39	4	0	1	83	4	0	1	127	4	0	1	171	8	0	1
40	4	0	1	84	4	0	1	128	4	0	1	172	4	0	1
41	4	0	1	85	4	0	1	129	4	0	1	173	4	0	1
42	4	0	1	86	4	0	1	130	4	0	1	174	4	0	1
43	4	0	1	87	4	0	1	131	4	0	1	175	4	0	1
44	4	0	1	88	4	0	1	132	4	0	1	176	4	0	1

Table A-2: Penalty weights for all models (cont’d)

Model	*α*	*β*	*δ*	Model	*α*	*β*	*δ*	Model	*α*	*β*	*δ*	Model	*α*	*β*	*δ*
177	4	0	1	221	4	0	2	285	4	0	1	329	4	0	1
178	4	0	1	222	4	0	1	286	4	0	1	330	4	0	1
179	4	0	1	223	4	0	1	287	4	0	1	331	4	0	1
180	4	0	1	224	4	0	1	288	4	0	1	332	4	0	1
181	4	0	2	225	4	0	1	289	4	0	1	333	4	0	1
182	4	0	1	226	4	0	1	290	4	0	1	334	4	0	1
183	4	0	1	227	4	0	1	291	4	0	1	335	4	0	1
184	4	0	1	228	4	0	1	292	4	0	1	336	4	0	1
185	4	0	1	229	4	0	1	293	4	0	1	337	4	0	1
186	4	0	1	230	4	0	1	294	4	0	1	338	4	0	1
187	4	0	1	231	4	0	1	295	4	0	1	339	4	0	1
188	4	0	1	232	4	0	1	296	4	0	1	340	4	0	1
189	4	0	1	233	4	0	1	297	4	0	1	341	4	0	1
190	4	0	1	234	4	0	1	298	4	0	1	342	4	0	1
191	4	0	1	235	4	0	1	299	4	0	1	343	4	0	1
192	4	0	1	236	4	0	1	300	4	0	1	344	4	0	1
193	4	0	1	237	4	0	1	301	6	0	1	345	4	0	1
194	4	0	1	238	4	0	1	302	4	0	1	346	4	0	1
195	4	0	1	239	4	0	1	303	4	0	1	347	4	0	1
196	4	0	1	240	4	0	1	304	4	0	1	348	4	0	1
197	4	0	1	241	4	0	1	305	4	0	1	349	4	0	1
198	4	0	1	242	4	0	1	306	4	0	1	350	4	0	1
199	4	0	1	243	4	0	1	307	4	0	1	351	4	0	1
200	4	0	1	244	4	0	1	308	4	0	1	352	4	0	1
201	4	0	2	245	4	0	1	309	4	0	1	353	4	0	1
202	4	0	1	246	4	0	1	310	4	0	1	354	4	0	1
203	4	0	1	247	4	0	1	311	4	0	1	355	4	0	1
204	4	0	1	248	4	0	1	312	4	0	1	356	4	0	1
205	4	0	1	249	4	0	1	313	4	0	1	357	4	0	1
206	4	0	1	250	4	0	1	314	4	0	1	358	4	0	1
207	4	0	1	251	4	0	1	315	4	0	1	359	4	0	1
208	4	0	1	252	4	0	1	316	4	0	1	360	4	0	1
209	4	0	1	253	4	0	1	317	4	0	1	361	6	0	1
210	4	0	1	254	4	0	1	318	4	0	1	362	4	0	1
211	4	0	4	255	4	0	1	319	4	0	1	363	4	0	1
212	4	0	1	256	4	0	1	320	4	0	1	364	4	0	1
213	4	0	1	257	4	0	1	321	4	0	2	365	4	0	1
214	4	0	1	258	4	0	1	322	4	0	1	366	4	0	1
215	4	0	1	259	4	0	1	323	4	0	1	367	4	0	1
216	4	0	1	260	4	0	1	324	4	0	1	368	4	0	1
217	4	0	1	281	4	0	2	325	4	0	1	369	4	0	1
218	4	0	1	282	4	0	1	326	4	0	1	370	4	0	1
219	4	0	1	283	4	0	1	327	4	0	1	371	6	0	1
220	4	0	1	284	4	0	1	328	4	0	1	372	4	0	1

Table A-2: Penalty weights for all models (cont'd)

Model	***α***	***β***	***δ***	**Model**	***α***	***β***	***δ***	**Model**	***α***	***β***	***δ***	**Model**	***α***	***β***	***δ***
373	4	0	1	382	4	0	1	391	4	0	1	400	4	0	1
374	4	0	1	383	4	0	1	392	4	0	1	Angel	4	0	2
375	4	0	1	384	4	0	1	393	4	0	1	Armadillo	4	0	2
376	4	0	1	385	4	0	1	394	4	0	1	Bunny	4	100	2
377	4	0	1	386	4	0	1	395	4	0	1	Canyon	4	100	2
378	4	0	1	387	4	0	1	396	4	0	1	Dinosaur	4	0	2
379	4	0	1	388	4	0	1	397	4	0	1	Dragon	4	0	2
380	4	0	1	389	4	0	1	398	4	0	1	Horse	4	0	1
381	4	0	1	390	4	0	1	399	4	0	1	Turbine	4	0	2

Table A-3 displays the root-mean-square average of the luminance differences between a representative rendering of each original model and its reduced versions.

Table A-3: RMS of luminance differences of models simplified with QEM (top) and our algorithm (bottom)

Model	50%	20%	10%	5%	2%	1%
1	0.020873	0.026478	0.033551	0.044898	0.066904	0.076304
	0.010394	0.023131	0.045521	0.056132	0.070847	0.081121
2	0.015726	0.018701	0.040625	0.029208	0.042938	0.054314
	0.009920	0.014711	0.038530	0.043373	0.041917	0.060596
3	0.007333	0.018175	0.025340	0.040738	0.058545	0.080211
	0.037244	0.021422	0.035000	0.065657	0.087449	0.095493
4	0.019127	0.025637	0.025351	0.036422	0.055449	0.074457
	0.037164	0.041833	0.040452	0.054015	0.076711	0.092756
5	0.009988	0.019871	0.026991	0.033823	0.047995	0.062093
	0.009509	0.022244	0.027771	0.038274	0.058609	0.083976
6	0.009095	0.040277	0.021532	0.031795	0.046401	0.058032
	0.009728	0.018993	0.028329	0.039344	0.060016	0.075309
7	0.011710	0.018282	0.023809	0.031727	0.043981	0.051924
	0.012753	0.017609	0.025225	0.051491	0.065027	0.087141
8	0.017151	0.016009	0.027137	0.023601	0.034521	0.042624
	0.005866	0.038247	0.041192	0.043176	0.056666	0.064430
9	0.035529	0.043209	0.047554	0.054941	0.067162	0.078566
	0.026535	0.032985	0.042745	0.063882	0.078221	0.092532
10	0.036529	0.039303	0.049056	0.046728	0.057833	0.073806
	0.036831	0.017355	0.043623	0.049667	0.060568	0.073716
11	0.004446	0.014234	0.022972	0.030997	0.044627	0.053639
	0.036659	0.018263	0.027915	0.055099	0.082625	0.095777
12	0.006495	0.026676	0.033337	0.037133	0.057027	0.066340
	0.030173	0.033718	0.035522	0.048517	0.066757	0.087732
13	0.040061	0.042571	0.042465	0.047231	0.046462	0.057257
	0.017786	0.023913	0.021797	0.032060	0.065386	0.079320
14	0.039155	0.042377	0.046132	0.051522	0.063473	0.074418
	0.039534	0.025710	0.035295	0.053133	0.102934	0.110237
15	0.016762	0.026495	0.037483	0.047541	0.065259	0.089644
	0.015164	0.029170	0.043003	0.061445	0.095779	0.098997
16	0.009791	0.020733	0.027909	0.038393	0.051392	0.062557
	0.028560	0.032526	0.038489	0.045337	0.060508	0.073890
17	0.009428	0.021043	0.027712	0.032192	0.039916	0.051270
	0.036813	0.019768	0.026383	0.035620	0.050659	0.070242
18	0.009543	0.019784	0.027138	0.034785	0.059943	0.069782
	0.036321	0.039848	0.045666	0.054031	0.066937	0.083905
19	0.008173	0.016131	0.022678	0.046968	0.045646	0.053006
	0.008004	0.021012	0.038474	0.046114	0.060420	0.076867

Table A-3: RMS of luminance differences of models simplified with QEM (top) and our algorithm (bottom) (cont'd)

Model	50%	20%	10%	5%	2%	1%
20	0.036894	0.032575	0.041676	0.044968	0.052620	0.060255
	0.032197	0.032066	0.043389	0.046370	0.063905	0.077155
21	0.012165	0.037408	0.039241	0.039437	0.035735	0.059793
	0.036447	0.037244	0.015258	0.021540	0.054924	0.064797
22	0.005154	0.012239	0.019534	0.025672	0.045847	0.067375
	0.032940	0.038926	0.041619	0.053864	**0.127741**	**0.141894**
23	0.018683	0.009402	0.014842	0.021358	0.036113	0.050105
	0.013848	0.009311	0.040490	0.021388	0.048817	0.048488
24	0.038390	0.039788	0.040935	0.045004	0.040209	0.068424
	0.006426	0.012015	0.043259	0.052148	**0.125964**	**0.145917**
25	0.035168	0.036251	0.037950	0.045332	0.048982	0.054134
	0.028968	0.036220	0.037720	0.039924	0.049258	0.056001
26	0.011389	0.015326	0.039928	0.027706	0.039448	0.054486
	0.006167	0.011136	0.025179	0.027371	0.038871	0.053252
27	0.004992	0.037277	0.015318	0.023071	0.042358	0.051950
	0.006172	0.020706	0.024162	0.026597	0.040687	0.059938
28	0.040849	0.041753	0.043691	0.048225	0.054345	0.059327
	0.013444	0.019932	0.019476	0.025497	0.037327	0.051522
29	0.003874	0.009256	0.014052	0.019900	0.031722	0.046328
	0.037190	0.035505	0.039043	0.019702	0.030910	0.057061
30	0.019780	0.035495	0.033888	0.021118	0.034486	0.050008
	0.036179	0.013104	0.016448	0.021084	0.028238	0.049935
31	0.021345	0.018730	0.022222	0.028374	0.044991	0.060361
	0.017012	0.034456	0.022169	0.025274	0.040067	0.048828
32	0.018387	0.034029	0.036208	0.040466	0.035691	0.051027
	0.015966	0.034329	0.035901	0.039049	0.036653	0.058001
33	0.018813	0.027598	0.034518	0.041075	0.065680	0.095086
	0.007253	0.015661	0.044594	0.057706	0.077472	0.124286
34	0.018284	0.010841	0.023674	0.024475	0.036682	0.048344
	0.005819	0.010722	0.016980	0.044111	0.040964	0.055527
35	0.003064	0.007166	*0.010716*	0.017005	0.030911	0.036368
	0.003791	0.007333	*0.011229*	0.016313	*0.026060*	*0.032386*
36	0.003489	0.013860	0.049525	0.053609	0.062598	0.068085
	0.038175	0.043453	0.018830	0.047984	0.046860	0.055160
37	0.027581	0.032065	0.026954	0.044723	0.092426	**0.125105**
	0.037213	0.041658	0.033928	0.053930	0.095881	0.133079
38	0.006107	0.015256	0.019579	0.034157	0.045602	0.055418
	0.007367	0.011641	**0.138536**	**0.145130**	0.095780	0.106114
39	0.036044	0.036815	0.038415	0.041251	0.050003	0.057650
	0.036176	0.037741	0.041021	0.047130	0.067578	**0.156325**
40	0.017794	0.018970	0.020862	0.020084	0.026035	0.037410
	0.013773	0.015049	0.016579	0.017318	*0.019050*	*0.034728*

Table A-3: RMS of luminance differences of models simplified with QEM (top) and our algorithm (bottom) (cont'd)

Model	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
41	0.055345	**0.064009**	0.054935	0.064783	0.053176	0.052146
	0.053360	0.052935	0.054213	0.056937	0.056086	0.056828
42	0.015681	0.017428	0.020385	0.038211	0.042100	0.038193
	0.033927	0.018650	0.019569	0.041121	0.066769	0.061894
43	0.003797	0.009106	0.012859	0.016251	0.022876	0.031740
	0.004290	0.010787	0.018227	0.026265	0.038437	0.046978
44	0.012403	0.038098	0.040636	0.026656	0.034698	0.050991
	0.037019	0.038674	0.041677	0.045268	0.038653	0.082408
45	0.007302	0.017317	0.023146	0.027280	0.035208	0.040930
	0.005988	0.012310	0.019180	0.030151	0.045940	0.054097
46	0.005597	0.016410	0.018230	0.024397	0.038036	0.034086
	0.004852	0.010777	0.016685	0.024544	0.035726	0.041812
47	0.005497	0.037764	0.019277	0.028647	0.039494	0.048442
	0.006500	0.017442	0.020272	0.029706	0.046854	0.063371
48	0.003141	0.008790	0.013503	0.025932	0.033074	0.040525
	0.013755	0.034562	0.036701	0.021113	0.066372	0.068417
49	0.034827	0.036480	0.016839	0.021425	0.035058	0.044500
	0.012384	0.020412	0.036862	0.028784	0.044369	0.055347
50	0.012790	0.014599	0.017126	0.020824	0.036916	0.028274
	0.013039	0.026984	0.029240	0.033178	0.038574	*0.040528*
51	0.006180	0.018730	0.022222	0.028374	0.044991	0.060361
	0.017012	0.034456	0.022169	0.025274	0.040067	0.048828
52	0.007165	0.015988	0.017300	0.022405	0.032223	0.041160
	0.007325	0.017015	0.025947	0.032566	0.041392	0.056097
53	0.017796	0.020671	0.024689	0.030961	0.035812	0.040672
	0.017647	0.021347	0.024753	0.027715	0.038703	0.048025
54	0.023124	0.024976	0.025422	0.030599	0.041127	0.045103
	0.024182	0.027452	0.024797	0.033424	0.040099	0.052242
55	**0.098418**	**0.099455**	**0.099032**	**0.097504**	0.091511	0.091907
	0.099501	**0.098685**	**0.098400**	**0.097459**	0.105997	0.103130
56	0.004217	0.011505	0.019587	0.027189	0.039205	0.049271
	0.006234	0.013336	0.041594	0.047805	0.051056	0.058754
57	0.010961	0.017745	0.020019	0.028412	0.029442	0.037140
	0.032147	0.025997	0.035429	0.025432	0.047237	0.048394
58	0.019672	0.029731	0.037441	0.047220	0.062400	0.072327
	0.013635	0.026638	0.036841	0.052830	0.075004	0.095466
59	0.013509	0.017735	0.025660	0.029415	0.040487	0.049961
	0.012799	0.023204	0.027596	0.030368	0.043297	0.052989
60	0.006460	0.022868	0.022456	0.030279	0.040113	0.044324
	0.006794	0.039550	0.022698	0.028849	0.042755	0.053602
61	0.004276	0.021572	0.039456	0.036961	0.045406	0.052572
	0.006123	0.012082	0.034114	0.035530	0.040334	0.053991

Table A-3: RMS of luminance differences of models simplified with QEM (top) and our algorithm (bottom) (cont'd)

Model	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
62	0.012793	0.015820	0.019560	0.025419	0.037240	0.052021
	0.013855	0.025424	0.020538	0.089241	0.047395	0.057483
63	0.036358	0.038918	0.015853	0.024808	0.029696	0.045907
	0.004198	0.014269	0.015668	0.024880	0.039440	0.066106
64	0.003257	0.010801	0.017358	0.024155	0.035874	0.051914
	0.004629	0.009647	0.023256	0.029559	0.042865	0.051933
65	0.036448	0.009677	0.013940	0.028572	0.038982	0.044470
	0.019173	0.037338	0.039250	0.042791	0.039639	0.050065
66	0.018147	0.036354	0.013129	0.028394	0.030534	0.039298
	0.036240	0.014232	0.018361	0.025630	0.037083	0.046046
67	*0.002303*	0.008960	0.014640	0.040792	0.031525	0.049181
	0.004377	0.011192	0.014979	0.022996	0.037147	0.046982
68	0.010687	0.033613	0.036737	0.027295	0.036136	0.048629
	0.012747	0.015427	0.020630	0.027930	0.039933	0.052851
69	0.011937	0.015099	0.015823	0.020373	0.036508	0.047638
	0.004586	0.010160	0.039977	0.045951	0.045324	0.087454
70	0.011590	0.010484	0.017243	0.023476	0.037704	0.047398
	0.004597	0.011208	0.018596	0.045425	0.044940	0.068743
71	0.017183	0.020577	0.019876	0.027594	0.041367	0.052429
	0.012198	0.037940	0.040753	0.046576	0.059564	0.067387
72	0.003301	0.010755	0.017775	0.029089	0.050706	0.066159
	0.041794	0.013628	0.040050	0.028334	0.064515	0.059559
73	0.037074	0.040091	0.018226	0.024841	0.040730	0.050133
	0.036561	0.016066	0.021557	0.031340	0.052497	0.048892
74	0.003423	0.010194	0.015731	0.026804	0.034517	0.063033
	0.032086	0.033450	0.040709	0.027587	0.042103	0.056211
75	0.003488	0.009098	0.014870	0.022657	0.031566	0.047957
	0.004798	0.037176	0.035838	0.022367	0.030243	0.056534
76	*0.002010*	0.036207	0.015142	0.022364	0.033096	0.043244
	0.004472	0.021017	0.025572	0.032424	0.041633	0.049541
77	0.003646	0.019544	0.024630	0.024984	0.039376	0.051084
	0.026691	0.011021	0.016701	0.026189	0.036713	0.052889
78	*0.002814*	0.009606	0.040008	0.038922	0.045265	0.060083
	0.005018	0.014026	0.025691	0.025686	0.050584	0.054996
79	0.003173	0.009494	0.016605	0.041500	0.038528	0.049716
	0.017693	0.010305	0.017007	0.024556	0.039225	0.060252
80	0.013372	0.018444	0.018788	0.023404	0.039411	0.051899
	0.004941	0.010313	0.016132	0.025703	0.039116	0.050970
81	0.008934	0.029518	0.032407	0.049212	0.064347	0.077778
	0.010703	0.021966	0.032911	0.049576	0.068737	0.088960
82	0.016389	0.028029	0.032816	0.045927	0.066745	0.081196
	0.038110	0.033356	0.045182	0.043435	0.060140	0.091319

Table A-3: RMS of luminance differences of models simplified with QEM (top) and our algorithm (bottom) (cont'd)

Model	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
83	0.007487	0.017535	0.043527	0.035373	0.052656	0.064941
	0.007828	0.040286	0.043969	0.036691	0.054784	0.074617
84	0.008281	0.017135	0.044046	0.036977	0.056184	0.071275
	0.037231	0.016267	0.043854	0.052946	0.062170	0.070875
85	0.021763	0.028652	0.031374	0.044385	0.064471	0.084664
	0.036663	0.040783	0.037448	0.050334	0.076169	0.105866
86	0.019146	0.023284	0.028592	0.036513	0.050275	0.068852
	0.020361	0.022694	0.032632	0.041936	0.063166	0.065637
87	0.035668	0.024459	0.023341	0.033340	0.053460	0.061710
	0.008874	0.015643	0.043168	0.049226	0.053338	0.084037
88	0.036837	0.042322	0.049134	0.047842	0.071726	0.087992
	0.009459	0.022299	0.037852	0.066844	0.084465	0.123601
89	0.028363	0.032376	0.037326	0.036149	0.053007	0.067121
	0.036083	0.038807	0.025979	0.037418	0.055082	0.070963
90	0.016006	0.041760	0.034319	0.045247	0.068400	0.092513
	0.013885	0.024013	0.035892	0.067631	0.086791	0.102254
91	0.032954	0.027573	0.034973	0.057357	0.079632	0.098087
	0.015510	0.026717	0.037968	0.058970	0.079920	0.102030
92	0.009828	0.022229	0.033098	0.044112	0.064247	0.077778
	0.011423	0.022325	0.048657	0.049766	0.072747	0.103566
93	0.015614	0.021578	0.027174	0.034687	0.049334	0.063774
	0.036725	0.018000	0.027953	0.044660	0.061714	0.082626
94	0.008574	0.041479	0.032684	0.048095	0.068422	0.096822
	0.015661	0.042342	0.049369	0.050045	0.085364	0.105506
95	0.011849	0.020828	0.030378	0.041550	0.058106	0.077435
	0.011369	0.020115	0.032875	0.051349	0.086359	0.117713
96	0.008996	0.017133	0.027747	0.038995	0.056430	0.069193
	0.037215	0.018165	0.027738	0.051417	0.070652	0.088745
97	0.015538	0.023692	0.031654	0.045170	0.066476	0.079567
	0.026236	0.036973	0.043783	0.047764	0.066057	0.093257
98	0.037021	0.041907	0.048371	0.046393	0.067120	0.073549
	0.037441	0.039317	0.034641	0.060036	0.085457	0.095085
99	0.005661	0.039621	0.021402	0.045961	0.046821	0.058114
	0.037370	0.039278	0.042462	0.029595	0.044876	0.057225
100	0.029817	0.022804	0.029718	0.037503	0.055261	0.068001
	0.019056	0.022723	0.025634	0.037753	0.052432	0.069756
101	0.009435	0.015421	0.022077	0.031176	0.044764	0.056546
	0.005475	0.015703	0.023057	0.030230	0.050370	0.060868
102	0.035168	0.029600	0.035115	0.044710	0.052993	0.073107
	0.026191	0.028504	0.032860	0.044045	0.055477	0.069380
103	0.002987	0.009525	0.017250	0.025106	0.036686	0.046566
	0.005423	0.011049	0.025418	0.027738	0.042436	0.061142

Table A-3: RMS of luminance differences of models simplified with QEM (top) and our algorithm (bottom) (cont'd)

Model	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
104	0.006625	*0.006277*	0.022537	0.026391	0.037739	0.051584
	0.019055	0.020257	0.024390	0.035141	0.058033	0.086016
105	0.008716	0.008261	0.028909	0.026799	0.040529	0.052607
	0.010031	0.013924	0.024200	0.049348	0.076242	N/A*
106	0.027250	0.028225	0.018407	0.024734	0.043907	0.054794
	0.005926	0.009469	0.030948	0.027125	0.049831	0.084974
107	0.014269	0.017960	0.026499	0.034021	0.049881	0.065856
	0.015242	0.027213	0.032184	0.035139	0.056018	0.068440
108	0.035269	0.023126	0.022281	0.030079	0.043395	0.055171
	0.013793	0.018248	0.026648	0.046161	0.056395	0.070811
109	*0.002313*	0.033837	0.037214	0.044405	0.041900	0.050915
	0.005355	0.012823	0.020841	0.035991	0.053714	0.099765
110	0.036224	0.012264	0.029096	0.069767	0.047923	0.067233
	0.012782	0.019626	0.034146	0.066182	0.111557	N/A*
111	0.010483	0.021599	0.029541	0.029867	0.040911	0.060546
	0.017795	0.018780	0.031116	0.087560	0.104379	0.104466
112	0.028395	0.012973	0.038298	0.023648	0.033357	0.044389
	0.005699	0.010907	0.017931	0.046796	0.071849	0.099822
113	0.009499	0.006659	0.020466	0.018542	0.029497	0.041277
	0.035457	0.038171	0.019843	0.036775	0.095770	0.088165
114	0.021002	0.016596	0.022498	0.029842	0.038542	0.044022
	0.009696	0.016267	0.027234	0.071246	0.098583	0.091111
115	0.012025	0.034613	0.024013	0.028704	0.042244	0.051080
	0.005021	0.021341	0.028172	0.062393	0.110413	0.113044
116	0.032665	0.032935	0.034393	0.026176	0.037398	0.045094
	0.016482	0.017685	0.022738	0.039101	0.092916	0.116432
117	0.035700	0.035757	0.036419	0.021913	0.029600	0.049443
	0.014609	0.036367	0.038843	0.045864	0.087399	0.092919
118	0.027670	0.021704	0.027713	0.035461	0.047863	0.057173
	0.018912	0.021820	0.023957	0.032797	0.054623	0.097614
119	0.005851	*0.006492*	0.038565	0.041238	0.051112	0.058415
	0.009865	0.009484	0.021119	0.045396	0.086164	0.109732
120	0.003399	0.019344	0.022638	0.029936	0.044283	0.055403
	0.019654	0.021684	0.027313	0.046477	**0.127678**	0.128122
121	0.009057	0.017788	0.027887	0.039473	0.059514	0.081618
	0.018713	0.046716	0.052885	0.047204	0.070719	0.103546
122	0.029471	0.035564	0.040708	0.058412	0.081204	**0.113618**
	0.025984	0.032783	0.043333	0.054602	0.087590	0.113636
123	0.003629	0.019887	0.022147	0.028988	0.043085	0.050994
	0.019604	0.021356	0.024095	0.045624	0.046720	0.065535
124	0.011888	0.034224	0.034119	0.040269	0.049917	0.052846
	0.038128	0.045088	0.043103	0.060275	0.069411	0.069549

Table A-3: RMS of luminance differences of models simplified with QEM (top) and our algorithm (bottom) (cont'd)

Model	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
125	0.019351	0.038551	0.054716	0.059799	0.082051	0.094118
	0.026888	**0.062051**	**0.084582**	**0.099320**	0.114330	0.166166
126	0.021761	0.029547	0.034466	0.044627	0.070231	0.082393
	0.038430	0.023169	0.050136	0.060099	0.086148	0.100499
127	0.035580	0.038055	0.043299	0.045806	0.072503	0.086346
	0.020468	0.025577	0.044174	0.052524	0.080671	0.103370
128	0.033473	0.038522	0.044077	0.051884	0.070574	0.086241
	0.024077	0.030187	0.036392	0.046863	0.073652	0.087279
129	0.036267	0.039470	0.043467	0.051238	0.063378	0.069982
	0.041783	0.044555	0.049885	0.057617	0.076729	0.094098
130	0.037607	0.023654	0.040205	0.053227	0.066885	0.080165
	0.037933	0.031422	0.037543	0.042697	0.070986	0.083427
131	0.007745	0.020367	0.032549	0.045258	0.071615	0.102075
	0.014691	0.024522	0.035115	0.051038	0.086626	0.125124
132	0.011562	0.022805	0.032311	0.047471	0.069759	0.090845
	0.036788	0.041672	0.054889	0.076850	0.105003	**0.138580**
133	0.010648	0.020352	0.031800	0.046706	0.074031	0.095504
	0.016010	0.022904	0.037265	0.057364	0.105091	**0.175492**
134	0.040487	0.037672	0.051630	0.065181	**0.095680**	**0.120892**
	0.027402	0.026683	0.038752	0.068643	0.100290	**0.142885**
135	0.010009	0.029717	0.038841	0.060643	0.085125	0.100023
	0.035086	0.027983	0.035990	0.063987	0.081859	0.116397
136	0.030827	0.037645	0.017850	0.029121	0.053000	0.055540
	0.036742	0.012359	0.041266	0.032665	0.050744	0.065172
137	0.036411	0.036373	0.030275	0.048403	0.059271	0.068792
	0.028151	0.036081	0.043172	0.047852	0.063092	0.076268
138	0.036434	0.040190	0.044844	0.048366	0.065608	0.081947
	0.032954	0.036266	0.041730	0.052939	0.075047	0.111684
139	0.035974	0.040182	0.046730	0.056796	0.070056	0.091818
	0.036516	0.040755	0.048028	0.057468	0.078934	0.110005
140	0.034874	0.034072	0.026382	0.050242	0.064292	0.083205
	0.035253	0.038409	0.035845	0.045185	0.061939	0.080492
141	*0.000408*	*0.003712*	0.039758	0.015593	0.031320	0.022035
	0.033460	0.037042	*0.008170*	*0.012164*	0.036578	0.051351
142	*0.000074*	0.015416	0.020659	0.030004	0.047478	0.058657
	0.031647	0.015149	0.019141	0.029100	0.052737	0.067001
143	0.027652	0.018739	0.016892	0.023906	0.034163	0.052359
	0.036578	0.036514	0.012690	0.041739	0.047120	0.062273
144	0.015407	0.015390	0.017915	0.025700	0.034807	0.047380
	0.014887	0.014923	0.024267	0.023853	0.038661	0.061775
145	0.009281	0.012179	0.013188	0.033505	0.045982	0.051629
	0.024973	0.024110	0.025644	0.019338	0.045128	0.046036

Table A-3: RMS of luminance differences of models simplified with QEM (top) and our algorithm (bottom) (cont’d)

Model	50%	20%	10%	5%	2%	1%
146	**0.064492**	0.054494	0.053841	0.058037	0.063506	0.071698
	0.053132	0.053102	0.065239	0.055868	0.060149	0.066807
147	0.029022	0.029624	0.016391	0.022159	0.026600	0.033679
	0.015395	0.014131	0.029382	0.021678	0.033151	0.051184
148	*0.000341*	0.008016	0.013398	0.021254	0.037848	0.047890
	0.004496	0.010699	0.013782	0.027738	0.042866	0.054224
149	0.004663	*0.006046*	*0.008654*	*0.012917*	0.019954	0.028834
	0.035375	0.005097	0.014381	0.038077	0.041988	0.053020
150	0.031948	0.033097	0.034811	0.021791	0.032151	0.038960
	0.017569	0.033333	0.034846	0.037024	0.043665	0.055019
151	0.033317	0.022129	0.024564	0.028285	0.035192	0.040913
	0.021394	0.022170	0.024269	0.023616	0.035655	0.043156
152	0.013483	0.036035	0.038119	0.044514	0.057050	0.069951
	0.028589	0.028590	0.031865	0.036857	*0.025044*	0.047730
153	0.007982	0.008405	*0.011294*	*0.015328*	0.026824	0.031244
	0.032288	0.032390	0.033019	0.034487	0.042565	0.048867
154	0.035889	0.037356	0.032531	0.043085	0.049089	0.058248
	0.032263	0.033384	0.034890	0.037913	0.045588	0.055264
155	*0.000179*	*0.004780*	0.012298	0.040603	0.027977	0.039741
	0.036079	0.036561	0.038138	0.040842	0.037235	0.058673
156	0.053529	0.018301	0.024030	0.028735	0.036103	0.039222
	0.016425	0.023315	0.023060	0.028221	0.038217	0.047422
157	0.028356	0.025464	0.029092	0.036385	0.042807	0.049798
	0.028855	0.029693	0.031834	0.035473	0.045453	0.072846
158	*0.001046*	*0.006024*	0.014946	0.018050	0.022496	0.029384
	0.035592	0.006315	0.038118	0.039898	0.034233	*0.037999*
159	0.030068	0.031472	0.033701	0.038386	0.046559	0.047460
	0.037370	0.038497	0.039700	0.041791	0.042930	0.049853
160	0.023923	0.014186	*0.009180*	*0.012754*	*0.015386*	0.021242
	0.019720	0.032504	0.019139	0.032852	0.050815	N/A*
161	0.030460	0.015234	0.039903	0.028244	0.048046	0.061454
	0.014327	0.017089	0.022559	0.030330	0.047182	0.067411
162	0.012685	0.016385	0.021234	0.032002	0.046525	0.062563
	0.013220	0.016503	0.021088	0.037857	0.046949	0.067784
163	0.004347	0.011135	0.019672	0.048401	0.049063	0.063655
	0.021198	0.011454	0.019364	0.030622	0.048249	0.064421
164	0.028104	0.036658	0.032295	0.049462	0.054332	0.069624
	0.036737	0.028777	0.039227	0.044705	0.057575	0.066194
165	0.032331	0.036376	0.039280	0.065179	0.077096	0.068294
	0.034892	0.019030	0.023889	0.031602	0.049216	0.066572
166	0.019591	0.026486	0.026418	0.035967	0.050299	0.062236
	0.021263	0.016372	0.030906	0.036773	0.051831	0.062780

Table A-3: RMS of luminance differences of models simplified with QEM (top) and our algorithm (bottom) (cont'd)

Model	50%	20%	10%	5%	2%	1%
167	0.036755	0.037884	0.040134	0.040034	0.055335	0.069958
	0.005631	0.010643	0.017138	0.043953	0.054353	0.064080
168	0.020096	0.010367	0.017881	0.031851	0.046333	0.059204
	0.021239	0.022937	0.037275	0.032333	0.047043	0.064854
169	0.036170	0.034060	0.037019	0.041217	0.057212	0.069331
	0.006032	0.011106	0.040994	0.045099	0.058966	0.073080
170	0.009635	0.013296	0.018430	0.027367	0.046245	0.062118
	0.006830	0.011724	0.020709	0.027696	0.043011	0.062168
171	0.027667	0.029276	0.032377	0.027203	0.041529	0.065206
	0.005073	0.013219	0.017812	0.024224	0.039437	0.062373
172	0.025209	0.028003	0.032498	0.063964	0.077437	0.072353
	0.015435	0.027442	0.031284	0.035490	0.054773	0.071337
173	0.036908	0.031728	0.033841	0.039764	0.051827	0.062275
	0.029802	0.031299	0.033971	0.039328	0.049480	0.066740
174	0.027925	0.015027	0.021191	0.029870	0.046588	0.067778
	0.036407	0.037602	0.017270	0.026719	0.040573	0.057336
175	0.032649	0.034422	0.038099	0.036577	0.060677	0.079772
	0.032418	0.021982	0.038126	0.045658	0.049292	0.073931
176	0.004263	0.012857	0.021920	0.031394	0.052897	0.075493
	0.036582	0.012443	0.018531	0.029055	0.047587	0.070382
177	0.037188	0.038408	0.041694	0.035808	0.054207	0.068132
	0.036746	0.017616	0.040465	0.045900	0.057649	0.062739
178	0.031059	0.032357	0.041006	0.048918	0.059944	0.074226
	0.032864	0.034181	0.044910	0.048617	0.052503	0.059927
179	0.027208	0.028928	0.018588	0.026766	0.043403	0.071765
	0.036801	0.037706	0.039443	0.026028	0.050298	0.063448
180	0.035957	0.010417	0.017371	0.026303	0.037218	0.057330
	0.005595	0.011174	0.016730	0.045436	0.053866	0.069775
181	0.036037	0.038161	0.034331	0.047986	0.058971	0.067669
	0.024867	0.033081	0.042475	0.051786	0.066245	0.074110
182	0.029164	0.041164	0.029872	0.034416	0.052383	0.057957
	0.027182	0.041683	0.043990	0.038801	0.058995	0.073190
183	0.036788	0.021751	0.022721	0.037877	0.059405	0.083643
	0.037164	0.039662	0.044175	0.052496	0.059866	0.095613
184	0.034237	0.038911	0.045382	0.060248	0.085937	0.108878
	0.029743	0.035841	0.039945	0.057045	0.079764	0.098131
185	0.031429	0.038416	0.046870	0.057608	0.070649	0.089549
	0.031787	0.041285	0.051026	0.064280	0.080630	0.088273
186	0.020902	0.024387	0.042774	0.050930	0.063359	0.071313
	0.008112	0.024731	0.030646	0.041022	0.068601	0.083812
187	0.036179	0.038250	0.042277	0.049201	0.059528	0.073478
	0.007701	0.014377	0.042518	0.048223	0.062894	0.078789

Table A-3: RMS of luminance differences of models simplified with QEM (top) and our algorithm (bottom) (cont'd)

Model	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
188	0.031559	0.033087	0.035573	0.039978	0.050271	0.050167
	0.027765	0.029144	0.026705	0.033900	0.051732	0.063762
189	0.027387	0.028656	0.025939	0.033791	0.045062	0.043634
	0.028290	0.029589	0.029977	0.035117	0.047335	0.051606
190	0.004290	0.013064	0.019012	0.029246	0.048015	0.057726
	0.006367	0.013493	0.020097	0.033845	0.048051	0.068220
191	0.024257	0.010207	0.017457	0.042435	0.053515	0.069319
	0.036364	0.037451	0.040177	0.044578	0.056449	0.071095
192	0.004500	0.039608	0.022393	0.044997	0.054187	0.073505
	0.007789	0.014012	0.024255	0.031045	0.051446	0.065521
193	0.014048	0.016642	0.023923	0.031440	0.046839	0.062265
	0.026796	0.029341	0.034009	0.041511	0.049160	0.065146
194	0.017525	0.023329	0.028599	0.034958	0.048903	0.059936
	0.013346	0.016757	0.026745	0.040491	0.050746	0.081421
195	0.032163	0.034096	0.037317	0.031616	0.046178	0.057431
	0.026939	0.024729	0.026632	0.034055	0.047672	0.063890
196	0.039637	0.012330	0.021919	0.036482	0.049494	0.061564
	0.036457	0.038679	0.041814	0.049417	0.059164	0.086099
197	0.007890	0.014903	0.020530	0.030215	0.044381	0.055281
	0.012618	0.013715	0.040435	0.031308	0.046035	0.060098
198	0.032797	0.028476	0.038330	0.041041	0.047816	0.055970
	0.031939	0.033004	0.038532	0.041129	0.044786	0.061166
199	0.018705	0.038137	0.034385	0.041651	0.064433	0.077479
	0.023879	0.022822	0.029782	0.039955	0.059085	0.086134
200	0.041021	0.029728	0.043709	0.058443	0.082808	0.109976
	0.022405	0.032584	0.047987	0.063091	0.093125	0.102363
201	0.036660	0.038571	0.042709	0.048694	0.064501	0.077726
	0.037060	0.039293	0.047153	0.053868	0.065956	0.071253
202	0.020731	0.024248	0.053616	0.035380	0.049351	0.070593
	0.033272	0.024465	0.030216	0.038743	0.052903	0.080115
203	0.006630	0.019059	0.025563	0.035052	0.053809	0.070735
	0.034424	0.020435	0.026895	0.035935	0.062826	0.087807
204	0.022604	0.023326	0.026479	0.039683	0.054178	0.069451
	0.028383	0.030460	0.034887	0.039541	0.055963	0.067989
205	0.039449	0.042396	0.045339	0.048204	0.066750	0.092365
	0.007691	0.035455	0.043616	0.051055	0.062256	0.085635
206	0.033879	0.035670	0.038233	0.047915	0.060762	0.078197
	0.037856	0.027352	0.032287	0.040828	0.059444	0.073200
207	0.049402	0.044973	0.045750	0.051409	0.049336	0.060143
	0.037041	0.022925	0.020641	0.037628	0.061677	0.076873
208	0.023553	0.026319	0.029835	0.035075	0.050630	0.073613
	0.021110	0.023530	0.027789	0.036910	0.054078	0.076246

Table A-3: RMS of luminance differences of models simplified with QEM (top) and our algorithm (bottom) (cont'd)

Model	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
209	0.032465	0.020286	0.024807	0.032244	0.047689	0.072874
	0.018027	0.020573	0.025651	0.033712	0.059178	0.075057
210	0.024229	0.044884	0.045730	0.049362	0.061061	0.071662
	0.032471	0.043279	0.041182	0.042182	0.056892	0.066848
211	0.037086	0.038865	0.041219	0.046551	0.052529	0.068941
	0.013473	0.014737	0.030497	0.035903	0.055705	0.073065
212	0.035315	0.039317	0.042537	0.048217	0.059141	0.072114
	0.035791	0.037810	0.040814	0.046314	0.066421	0.081316
213	0.005539	0.013236	0.020565	0.031212	0.060103	0.073954
	0.036129	0.014842	0.025299	0.035176	0.067314	0.082943
214	0.014118	0.018651	0.023931	0.037770	0.068647	0.078767
	0.007648	0.019113	0.024980	0.039461	0.058856	0.088586
215	0.020521	0.035916	0.037662	0.044819	0.054923	0.070000
	0.033348	0.028082	0.036644	0.045427	0.059825	0.078239
216	0.038173	0.040397	0.043566	0.050531	0.065168	0.079825
	0.025719	0.040736	0.044385	0.051445	0.062569	0.077296
217	0.019626	0.030536	0.034986	0.038175	0.064506	0.075035
	0.023367	0.025974	0.037314	0.040812	0.050559	0.076638
218	0.039898	0.052879	0.055747	0.055931	0.063131	0.076396
	0.019988	0.022662	0.026462	0.033382	0.047369	0.068750
219	0.023255	0.027866	0.032200	0.050117	0.073278	0.086501
	0.034877	0.025541	0.030803	0.052499	0.066922	0.097204
220	0.015339	0.014322	0.022617	0.029948	0.043685	0.066648
	0.013985	0.015048	0.024127	0.036566	0.070991	0.103982
221	0.039226	0.039820	0.041286	0.041737	0.054706	0.058086
	0.035494	0.036271	0.041774	0.047326	0.059364	0.077065
222	0.035392	0.037399	0.039130	0.024333	0.032436	0.054926
	0.036322	0.037270	0.016172	0.029418	0.057231	0.069374
223	0.017689	0.034993	0.038054	0.042398	0.056085	0.060297
	0.033733	0.031248	0.018356	0.025812	0.041627	0.066328
224	0.012577	0.010521	0.017849	0.028525	0.046878	0.063580
	0.005325	0.011144	0.031359	0.027472	0.044294	0.070199
225	0.011655	0.026358	0.020996	0.028515	0.042578	0.057797
	0.018046	0.020563	0.025264	0.028846	0.040750	0.054913
226	0.003746	0.022576	0.025600	0.030545	0.032051	0.045311
	0.004239	0.019335	0.022173	0.031787	0.053551	0.073464
227	0.014356	0.016836	0.020714	0.026584	0.032051	0.045311
	0.004239	0.012047	0.016215	0.027959	0.052119	0.074760
228	0.011541	0.013905	0.018186	0.024123	0.037705	0.047290
	0.021613	0.014135	0.019110	0.024042	0.044847	0.064378
229	0.012083	0.014996	0.019017	0.024375	0.036698	0.049939
	0.027579	0.015328	0.018753	0.024921	0.040991	0.068753

Table A-3: RMS of luminance differences of models simplified with QEM (top) and our algorithm (bottom) (cont’d)

Model	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
230	0.039479	0.033234	0.038626	0.042525	0.049935	0.061182
	0.023750	0.025131	0.038983	0.043274	0.055076	0.066569
231	0.036328	0.013881	0.022135	0.030532	0.054618	0.077891
	0.036405	0.038268	0.042351	0.031779	0.057832	0.074418
232	0.031097	0.034422	0.020050	0.036588	0.049217	0.068576
	0.036580	0.038410	0.021409	0.032553	0.066609	0.082209
233	0.013170	0.017110	0.025262	0.031108	0.044932	0.058897
	0.037367	0.020503	0.042603	0.033979	0.049411	0.062718
234	0.016995	0.027832	0.032145	0.031981	0.051698	0.061969
	0.025673	0.023134	0.032386	0.039232	0.056904	0.072037
235	0.018724	0.021136	0.032418	0.032675	0.047541	0.062227
	0.029306	0.031228	0.034221	0.033854	0.045408	0.073015
236	0.019369	0.023002	0.028392	0.036727	0.051361	0.073644
	0.014556	0.022884	0.027650	0.038529	0.057106	0.103989
237	0.005479	0.014178	0.020030	0.027763	0.045193	0.068000
	0.036611	0.038564	0.041639	0.032919	0.058485	0.103647
238	0.039802	0.039080	0.021847	0.038050	0.053450	0.064820
	0.012895	0.014440	0.020932	0.034674	0.066045	0.097447
239	0.004794	0.012448	0.020277	0.036414	0.045614	0.070397
	0.030642	0.032608	0.040889	0.029507	0.040977	0.064017
240	0.009288	0.015028	0.023878	0.031507	0.045821	0.067555
	0.012005	0.016648	0.022106	0.030772	0.054523	0.067659
241	0.038456	0.040242	0.043263	0.047241	0.053333	0.062068
	0.027459	0.030095	0.034183	0.047835	0.055393	0.060771
242	0.033979	0.037642	0.018294	0.024688	0.034425	0.043164
	0.005075	0.018699	0.017758	0.026066	0.038486	0.059393
243	0.020727	0.023855	0.028537	0.034341	0.059306	0.079307
	0.032458	0.034520	0.038526	0.043908	0.055797	0.075906
244	0.035601	0.038022	0.044821	0.048736	0.051726	0.055128
	0.028442	0.037728	0.042711	0.046855	0.059353	0.075512
245	0.028904	0.031189	0.032949	0.038562	0.044072	0.053796
	0.018486	0.030700	0.034333	0.038382	0.048473	0.061839
246	0.029896	0.039692	0.042365	0.046610	0.058646	0.067445
	0.038325	0.039860	0.033306	0.037370	0.048573	0.067989
247	0.011610	0.012957	0.021970	0.030194	0.042036	0.059776
	0.015347	0.019095	0.024370	0.027901	0.048558	0.067075
248	0.014313	0.017188	0.022355	0.029667	0.050821	0.053794
	0.035355	0.036358	0.021638	0.029466	0.042219	0.058524
249	0.005285	0.014871	0.023696	0.035220	0.052985	0.063999
	0.035877	0.025982	0.030719	0.039518	0.063645	0.085934
250	0.029115	0.036412	0.039231	0.042091	0.049885	0.058559
	0.043865	0.044997	0.048171	0.044483	0.052924	0.100629

Table A-3: RMS of luminance differences of models simplified with QEM (top) and our algorithm (bottom) (cont’d)

Model	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
251	0.024889	0.016552	0.022787	0.038361	0.053914	0.075360
	0.016264	0.024108	0.026245	0.033156	0.050028	0.075877
252	0.036966	0.038755	0.041126	0.041677	0.053019	0.063504
	0.037233	0.038930	0.041908	0.047435	0.054775	0.061590
253	0.037188	0.015072	0.041832	0.044907	0.052673	0.065326
	0.037460	0.015691	0.049694	0.033244	0.055607	0.073537
254	0.004071	0.020930	0.042007	0.025236	0.041969	0.042708
	0.012526	0.016561	0.019191	0.025665	0.040681	0.050438
255	0.033038	0.035291	0.038687	0.043299	0.056796	0.070561
	0.033345	0.035143	0.037898	0.035428	0.052879	0.079070
256	0.035380	0.035187	0.039359	0.042982	0.049972	0.056179
	0.035494	0.036772	0.018153	0.028189	0.054149	0.069215
257	0.041925	0.055093	0.064371	**0.076583**	**0.107687**	**0.135066**
	0.036531	0.047809	**0.083005**	**0.107692**	**0.119630**	0.124961
258	0.014133	0.022314	0.030859	0.048918	0.060210	0.072412
	0.014620	0.047034	**0.085867**	**0.103843**	**0.122634**	0.125800
259	0.005583	0.012301	0.017944	0.033969	0.043219	0.067418
	0.035562	0.037250	0.020275	0.033096	0.066221	0.077498
260	0.017371	0.032071	0.042577	0.053483	0.067971	0.079211
	0.020054	0.041677	0.048565	0.064029	0.084030	0.094424
281	0.020442	0.029167	0.031890	0.043215	0.058997	0.069868
	0.016303	0.026815	0.032724	0.043014	0.064682	0.084224
282	0.009027	0.019363	0.025594	0.033187	0.047130	0.056739
	0.010362	0.017814	0.025363	0.034514	0.052894	0.065253
283	0.039492	0.021058	0.025641	0.029390	0.038556	0.047985
	0.036979	0.039075	0.041748	0.046768	0.059830	0.071167
284	0.017559	0.032338	0.041223	0.051360	0.079555	0.086517
	0.019407	0.028843	0.040333	0.059800	0.113224	0.128247
285	0.037667	0.031793	0.027320	0.034979	0.047613	0.058715
	0.010232	0.040860	0.044453	0.049201	0.061565	0.075273
286	0.031319	0.021793	0.047540	0.038359	0.058069	0.064826
	0.032572	0.019976	0.044318	0.046009	0.062602	0.076812
287	0.015458	0.022949	0.026950	0.036047	0.050530	0.061267
	0.038274	0.042207	0.048080	0.062478	0.098610	0.107542
288	0.014604	0.024969	0.030381	0.038479	0.053374	0.064562
	0.011692	0.020073	0.026578	0.053668	0.057430	0.076068
289	0.010017	0.019810	0.025608	0.033436	0.047052	0.061476
	0.031323	0.040122	0.043816	0.049199	0.062360	0.068808
290	0.010454	0.043903	0.029129	0.038274	0.054660	0.064371
	0.013000	0.023424	0.036733	0.064383	**0.126677**	0.129734
291	0.016076	0.036669	0.033198	0.048916	0.061991	0.071980
	0.033053	0.038560	0.033452	0.042171	0.059850	0.076801

Table A-3: RMS of luminance differences of models simplified with QEM (top) and our algorithm (bottom) (cont'd)

Model	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
292	0.009251	0.041555	0.042342	0.032291	0.054376	0.053423
	0.010251	0.016110	0.022401	0.032485	0.046760	0.058034
293	0.019210	0.028568	0.047067	0.043447	0.058578	0.073460
	0.017982	0.024871	0.032641	0.045188	0.061261	0.077770
294	0.015269	0.020958	0.025323	0.030016	0.039157	0.046257
	0.016465	0.020925	0.025659	0.033915	0.049602	0.065659
295	0.011519	0.021529	0.046303	0.037318	0.051787	0.062647
	0.040093	0.042719	0.026727	0.036655	0.055141	0.068499
296	0.014162	0.036665	0.042731	0.050251	0.058057	0.068817
	0.040121	0.044301	0.048378	0.056652	0.064639	0.088544
297	0.013776	0.026103	0.035103	0.044250	0.065601	0.076755
	0.016064	0.023903	0.031953	0.042711	0.061187	0.082733
298	0.018941	0.025558	0.031475	0.038893	0.051498	0.065386
	0.019308	0.024444	0.031988	0.039047	0.057739	0.085005
299	0.014536	0.034304	0.036086	0.044882	0.064066	0.078654
	0.038875	0.044061	0.049568	0.057999	0.081252	0.087155
300	0.019749	0.023456	0.030486	0.040116	0.054311	0.067831
	0.013593	0.042591	0.047395	0.055741	0.069587	0.089204
301	0.033725	0.038016	0.041164	0.048194	0.056847	0.070306
	0.038671	0.040860	0.043740	0.041365	0.061001	0.077528
302	0.012131	0.023520	0.033391	0.043100	0.066386	0.083031
	0.031518	0.024177	0.033781	0.052205	0.085949	0.130768
303	0.039322	0.046278	0.051450	0.050074	0.070912	0.086197
	0.023849	0.039364	0.047707	0.051844	0.075833	0.090049
304	0.030050	0.032250	0.027917	0.026363	0.038598	0.049286
	0.038867	0.037953	0.040827	0.027639	0.046007	0.056231
305	0.011344	0.018001	0.025837	0.032712	0.042973	0.052568
	0.012623	0.019143	0.022839	0.030527	0.045317	0.057787
306	0.033274	0.026911	0.029340	0.033191	0.040691	0.055039
	0.033573	0.034607	0.036697	0.033107	0.038208	0.050480
307	0.016362	0.020890	0.029158	0.034995	0.043720	0.061554
	0.017137	0.014619	0.020720	0.033362	0.045817	0.060619
308	0.012623	0.015448	0.016601	0.022857	0.035136	0.050482
	0.013544	0.014441	0.038574	0.041148	0.035398	0.047049
309	0.030019	0.049724	**0.065597**	0.071097	0.091955	0.097089
	0.066223	**0.079898**	0.045054	0.050725	0.086897	0.118179
310	0.013247	0.038691	0.043925	0.042656	0.037797	0.032055
	0.047153	0.041861	0.061980	0.070382	0.077317	0.074089
311	0.007042	0.020318	0.033451	0.039887	0.054566	0.067626
	0.016578	0.040817	0.028684	0.040294	0.054165	0.065775
312	**0.056061**	0.039934	0.044073	0.036681	0.052565	0.065280
	0.075178	0.005972	0.015476	0.019954	*0.027879*	0.041233

Table A-3: RMS of luminance differences of models simplified with QEM (top) and our algorithm (bottom) (cont’d)

Model	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
313	0.015624	0.032660	0.044797	0.058313	0.078953	0.093628
	0.036557	0.045465	0.041929	0.054056	0.073364	0.092819
314	0.003547	0.011933	0.019593	0.030289	0.033519	0.042966
	0.037071	0.012441	0.019201	0.026185	0.039678	0.058905
315	0.004941	0.013380	0.021253	0.029951	0.045111	0.053167
	0.039610	0.041212	0.024417	0.032604	0.044987	0.057846
316	0.007001	0.026707	0.027024	0.035407	0.052525	0.071127
	0.014310	0.021705	0.029555	0.039310	0.051411	0.067905
317	0.004305	0.022539	0.042832	0.029065	0.042814	0.055844
	0.036557	0.038001	0.041284	0.045807	0.039252	0.054786
318	0.014566	0.019928	0.026089	0.047021	0.042081	0.052456
	0.032021	0.034322	0.041953	0.033309	0.047658	0.055522
319	0.038188	0.037554	0.038925	0.019896	0.034877	0.043660
	0.036697	0.037338	0.038666	0.040637	0.036196	0.047840
320	0.011930	0.021856	0.030812	0.039424	0.054865	0.071512
	0.015521	0.022030	0.035000	0.047872	0.062191	0.074565
321	0.033423	0.033428	0.034399	0.035503	0.018431	0.018449
	0.034251	0.034261	0.034256	0.034244	0.040195	0.078185
322	0.029137	0.029590	*0.010968*	*0.011026*	*0.011073*	*0.011289*
	0.009862	0.009764	0.012913	*0.009838*	0.031495	0.080610
323	0.010798	0.009822	*0.010018*	*0.010221*	*0.010238*	*0.010370*
	0.037133	0.032469	*0.009862*	*0.009930*	0.029404	0.058864
324	0.037399	0.037544	0.016345	0.038899	0.029584	0.034661
	0.034259	0.034390	0.035214	0.038782	0.063612	0.118004
325	0.032934	0.033596	0.015650	0.015626	0.015596	*0.015610*
	0.051248	0.016616	0.016748	*0.015424*	*0.022553*	0.094249
326	0.034589	0.016060	0.016646	0.034503	0.017840	0.017788
	0.015527	0.015704	0.016384	0.035405	0.056654	0.086593
327	0.025920	0.025934	0.028597	0.019865	0.021586	0.021737
	0.039699	0.035351	0.021415	0.023902	0.115884	0.119284
328	0.009716	0.037126	*0.005828*	*0.010521*	*0.010602*	*0.010719*
	0.010487	0.010487	*0.010490*	*0.008067*	0.032455	0.086779
329	0.034269	0.015521	0.017608	0.020434	0.017725	0.107499
	0.016379	0.018637	0.015768	*0.015505*	*0.016932*	*0.015501*
330	0.033675	0.015387	0.015387	*0.015387*	*0.015422*	*0.015461*
	0.051245	0.010986	*0.010987*	*0.013719*	*0.022516*	0.061507
331	**0.061502**	**0.063010**	0.063010	0.061504	0.061509	0.061504
	0.035166	0.035167	0.035167	0.033455	0.030674	0.087899
332	0.015375	0.018348	0.015798	0.015839	0.015943	0.018267
	0.033806	0.033817	0.021530	0.021667	0.031486	0.048657
333	**0.060152**	0.060162	**0.068405**	0.068606	0.057334	0.057342
	0.035749	0.035765	0.035767	0.035773	0.038152	0.102844

Table A-3: RMS of luminance differences of models simplified with QEM (top) and our algorithm (bottom) (cont'd)

Model	**50%**	**20%**	***10%***	**5%**	**2%**	**1%**
334	0.034452	0.034454	0.034457	0.033339	*0.015444*	*0.015493*
	0.033810	0.033808	0.033810	0.033830	0.062056	0.088876
335	0.036020	0.036029	0.036974	0.034178	0.017525	0.017547
	0.018644	0.019494	0.036073	0.036071	0.042031	0.136684
336	0.031627	0.031854	0.032485	0.017965	0.022386	0.024934
	0.032976	0.033056	0.033406	0.034367	0.036313	*0.039532*
337	0.003849	0.009695	0.021488	0.029421	0.038527	0.059027
	0.006706	0.012444	0.017421	0.027365	0.045752	0.067751
338	0.004666	0.013306	0.017334	0.020967	0.026755	0.031017
	0.025650	0.013534	0.038314	0.040999	0.036994	0.045491
339	**0.060440**	**0.061822**	**0.068721**	0.068491	0.061494	0.061940
	0.033723	0.033629	0.033613	0.033569	*0.008984*	0.105922
340	0.022163	0.011501	0.021982	0.015472	*0.014988*	*0.015525*
	0.033707	0.033706	0.033701	0.033741	0.076426	0.111734
341	0.003555	*0.006497*	0.015582	0.028401	0.055451	0.065758
	0.029057	**0.065742**	**0.086430**	0.088122	0.094099	0.096824
342	0.003660	*0.005574*	0.013168	0.024402	0.053644	0.061209
	0.024218	**0.071273**	0.078470	0.081077	0.086222	0.088369
343	0.036825	*0.005059*	0.012063	0.028150	0.050137	0.059590
	0.029160	0.048464	0.056093	0.068023	0.077166	0.083651
344	0.016774	0.014697	0.016462	*0.013974*	0.022426	0.034481
	0.036350	0.005427	0.018415	0.022308	0.049328	0.076655
345	0.017474	0.014318	0.036256	*0.014143*	0.026721	0.037521
	0.030021	0.007140	*0.012352*	0.036703	0.057068	0.078294
346	0.041571	0.020369	0.012159	0.018204	0.033587	0.045844
	0.036288	0.036748	*0.012544*	0.034426	0.060096	0.090781
347	0.022780	0.032564	0.034608	0.036906	0.046269	0.059915
	0.020296	0.022196	0.029298	0.056969	0.082133	0.074818
348	0.019078	0.019210	0.019505	0.020103	0.025881	0.030874
	0.035625	0.034223	0.040312	0.050348	0.051452	0.065942
349	0.021623	0.021916	*0.007433*	0.022598	0.028055	0.040521
	0.046251	**0.080327**	**0.084748**	**0.091138**	0.092877	0.098157
350	0.014591	0.020566	0.014898	0.015684	0.021196	0.025952
	0.018399	0.018561	0.019251	0.021788	0.035029	0.068934
351	**0.099964**	**0.122034**	**0.145123**	**0.150743**	**0.149662**	**0.148789**
	0.006093	0.013978	0.066741	N/A[*]	N/A[*]	N/A[*]
352	0.035514	*0.002272*	*0.003518*	*0.006488*	*0.010312*	*0.014555*
	0.013160	0.002761	0.035454	*0.005897*	0.036582	0.047959
353	**0.098041**	**0.098873**	**0.125329**	**0.126612**	**0.129216**	**0.130942**
	0.060939	**0.104302**	**0.106399**	**0.107001**	0.107397	0.108763
354	0.036563	0.007728	0.012401	0.019559	0.035361	0.042515
	0.006560	0.010990	0.019443	0.059044	0.062067	0.069873

Table A-3: RMS of luminance differences of models simplified with QEM (top) and our algorithm (bottom) (cont'd)

Model	50%	20%	10%	5%	2%	1%
355	0.035881	0.035589	0.035634	0.037065	*0.011763*	*0.011857*
	0.010975	0.011445	*0.011688*	*0.011812*	0.036132	*0.037055*
356	0.023784	0.017473	0.017564	0.031183	*0.014092*	*0.014329*
	0.025041	0.021982	0.022929	0.025293	*0.025467*	*0.027383*
357	0.036749	0.050686	0.046849	0.047587	0.036413	0.036413
	0.035716	0.035852	0.035853	0.036403	0.042991	0.068385
358	**0.115529**	**0.127988**	**0.133339**	**0.130313**	**0.125254**	**0.125258**
	0.013474	0.004765	0.036285	0.040724	0.086428	0.090747
359	0.021337	0.043221	0.045796	0.021767	0.029336	0.041042
	0.169677	0.005088	*0.005491*	*0.014987*	0.067794	0.075491
360	0.004997	**0.063902**	0.037336	0.027108	0.049055	0.061159
	0.008318	0.013503	0.023317	0.053048	0.067392	0.061392
361	0.014192	0.020194	0.032142	0.041208	0.064334	0.083246
	0.025607	0.025346	0.034771	0.045483	0.069192	0.095120
362	0.011218	0.015297	0.023938	0.017678	0.024637	0.032841
	0.019014	0.017784	0.020016	0.026824	0.032391	*0.041060*
363	0.003186	0.019585	*0.006026*	0.037842	0.021191	0.021614
	0.017021	0.017317	0.017907	0.018993	*0.023527*	*0.025203*
364	0.044246	0.037988	0.041562	0.055936	0.069391	0.086666
	0.044552	0.047310	0.033718	0.054004	0.077385	0.104757
365	0.035862	0.037886	0.019643	0.025955	0.039703	0.059451
	0.036874	0.038379	0.041084	0.045484	0.060452	0.092060
366	0.038342	0.037167	0.038751	0.042131	0.053589	0.067457
	0.036645	0.037028	0.038244	0.036864	0.051392	0.069675
367	0.020715	0.029680	0.038338	0.052387	0.066465	0.087319
	0.009401	0.017060	0.035625	0.051832	0.080835	0.111456
368	0.013640	0.013727	0.016831	0.028766	0.045812	0.057033
	0.037132	0.038453	0.040096	0.044041	0.055231	0.071258
369	0.007743	0.015254	0.020933	0.030081	0.048125	0.055161
	0.012252	0.014725	0.021923	0.033863	0.054235	0.079328
370	0.040757	0.041459	0.041663	0.043995	0.045371	0.058668
	0.023661	0.024524	0.041798	0.043445	0.049179	0.062298
371	0.029025	0.029409	0.040063	0.036928	0.044571	0.052034
	0.034349	0.028015	0.028965	0.034028	0.045884	0.046694
372	0.029849	0.038325	0.035320	0.040718	0.060899	0.072292
	0.036912	0.038259	0.035703	0.041564	0.065897	0.084785
373	*0.002933*	0.013998	0.012926	0.018400	0.030097	0.045100
	0.020809	0.007411	*0.011893*	0.018793	0.031921	0.044544
374	0.004654	0.013181	0.020216	0.025046	0.026517	0.031596
	0.036491	0.036847	0.015880	0.020305	0.029149	0.041398
375	0.036046	0.041263	0.021849	0.027921	0.038272	0.055114
	0.008500	0.022099	0.028936	0.026166	0.042244	0.070390

Table A-3: RMS of luminance differences of models simplified with QEM (top) and our algorithm (bottom) (cont’d)

Model	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
376	0.037890	0.019831	0.026650	0.036877	0.048805	0.073962
	0.010843	0.018190	0.026538	0.038288	0.059683	0.075852
377	0.036192	0.039732	0.022564	0.026800	0.041006	0.053846
	0.031554	0.036221	0.039033	0.030358	0.049707	0.066639
378	0.025766	0.031738	0.028878	0.037394	0.060051	0.079311
	0.013559	0.031907	0.032504	0.042939	0.064249	0.101953
379	0.038339	0.025393	0.029799	0.039215	0.054897	0.079790
	0.038173	0.039717	0.043752	0.044248	0.062579	0.092830
380	0.035271	0.036411	0.039118	0.042346	0.046920	0.064122
	0.035603	0.036588	0.038431	0.041227	0.056984	0.075035
381	0.017810	0.027090	0.034577	0.041604	0.060909	0.070302
	0.018791	0.021957	0.033607	0.052884	0.083574	0.104570
382	0.009857	0.022055	0.032701	0.044965	0.066789	0.091750
	0.011240	0.024846	0.037586	0.049391	0.083102	0.113167
383	0.039449	0.043620	0.027740	0.041480	0.065546	0.085802
	0.037465	0.041971	0.047685	0.059925	0.087317	0.102060
384	0.016709	0.032814	0.046728	0.062700	0.082563	0.104156
	0.018707	0.033810	0.047204	0.064633	0.097648	0.112693
385	0.024013	0.028725	0.034305	0.041667	0.060470	0.075048
	0.023614	0.021268	0.030283	0.046088	0.072050	0.094646
386	0.032765	0.056731	**0.070538**	**0.093188**	**0.109616**	**0.125969**
	0.033682	**0.057727**	**0.079221**	**0.122038**	**0.166601**	**0.164513**
387	0.028701	0.017406	0.024502	0.030900	0.047411	0.055359
	0.036617	0.039304	0.043122	0.034930	0.052733	0.072936
388	0.038862	0.024280	0.023834	0.033771	0.048734	0.066822
	0.008195	0.039458	0.044810	0.045672	0.067031	0.096332
389	0.023405	0.030347	0.035908	0.044602	0.057269	0.069628
	0.023793	0.029653	0.037153	0.048005	0.066058	0.070522
390	0.004386	0.012468	0.026146	0.031259	0.033696	0.042980
	0.023894	0.035228	0.021810	0.026012	0.043553	0.071985
391	0.040034	0.046122	0.052076	0.054710	0.074287	0.086228
	0.040370	0.043379	0.048020	0.063651	0.075765	0.112865
392	0.038457	0.014630	0.031490	0.033173	0.043447	0.064331
	0.017198	0.032059	0.037382	0.046816	0.067212	0.095152
393	0.031258	0.029566	0.033358	0.038830	0.053428	0.064344
	0.022994	0.043444	0.048576	0.055086	0.075791	0.096666
394	0.006977	0.016284	0.039943	0.035915	0.043194	0.060266
	0.007795	0.015825	0.031423	0.036778	0.057584	0.095898
395	**0.094308**	**0.114682**	**0.116990**	**0.123211**	**0.126083**	**0.128233**
	0.023710	0.030328	0.051266	0.070986	0.108379	0.112407
396	0.036722	0.022547	0.030471	0.044056	0.067894	0.082868
	0.037541	0.042352	0.047971	0.050204	0.093812	0.106416

Table A-3: RMS of luminance differences of models simplified with QEM (top) and our algorithm (bottom) (cont’d)

Model	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
397	0.014823	0.017081	0.027800	0.044083	0.054088	0.060429
	0.032805	0.035549	0.040401	0.051204	0.061278	0.070160
398	0.019700	0.044719	0.060138	**0.076765**	**0.104849**	0.105356
	0.045139	**0.053264**	**0.086974**	**0.124495**	**0.153122**	0.112469
399	0.021150	0.021828	0.030966	0.042424	0.060172	0.074959
	0.010991	0.020166	0.030869	0.054618	0.067030	0.080650
400	0.008576	0.018690	0.029387	0.039228	0.055594	0.066406
	0.032525	0.022718	0.032211	0.037369	0.059039	0.079814
Angel	0.014139	0.032132	0.032910	0.034025	0.032070	0.040792
	0.031598	0.032095	0.032787	0.034232	0.037597	0.042236
Armadillo	0.031939	0.030912	0.034436	0.037393	0.049770	0.057793
	0.016829	0.024094	0.035528	0.040600	0.045008	0.053709
Bunny	0.025511	0.041030	0.055987	**0.072644**	**0.094844**	0.110413
	0.025744	0.041697	0.057375	0.077030	**0.122551**	**0.160426**
Canyon	0.049364	**0.093497**	**0.111589**	**0.121958**	**0.141523**	**0.150134**
	0.085059	**0.115739**	0.035854	**0.135881**	**0.157200**	**0.167699**
Dinosaur	0.005684	0.009487	0.038467	0.019460	0.027968	0.040933
	0.039054	0.037134	0.038648	0.020738	0.035993	0.054517
Dragon	0.028082	0.033012	0.025692	0.044294	0.052977	0.063510
	0.013399	0.020140	0.027092	0.036105	0.056378	0.080200
Horse	0.040753	0.032168	0.034074	0.029448	0.050884	0.046243
	0.025982	0.027367	0.028554	0.022322	0.054128	0.064437
Turbine	0.027627	0.032332	0.036445	0.041952	0.052423	0.059264
	0.027614	0.033361	0.040413	0.054371	0.083463	0.097988
Average	0.022643	0.027375	0.031574	0.037908	0.049846	0.061883
	0.024873	0.027678	0.033843	0.042246	0.059792	0.077612
Max	0.115529	0.127988	0.145123	0.150743	0.149662	0.150134
	0.169677	0.115739	0.138536	0.145130	0.166601	0.175492
Min	0.000074	0.002272	0.003518	0.006488	0.010238	0.010370
	0.003791	0.002761	0.005491	0.005897	0.008984	0.015501

Bold indicates 10 highest RMS luminance difference averages at the given percentage for the given algorithm

Italics indicate 10 lowest values at the given percentage for the given algorithm

* N/A: The algorithm exhausted all possible contractions before the given level

Table A-4 displays the results for meshes with uneven vertex distribution that were created using QEM reduction to 50%, before performing further simplification with both algorithms. As this test has been performed using fewer meshes, we have chosen to display two sets of averages, one with all data, and the other with significant outliers (marked in italics) removed.

Table A-4: Hausdorff distances on 50% reduced meshes

Model	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
1	0.005891	0.011402	0.018834	0.022061	0.084247	0.111470
	0.004732	0.018146	0.030507	0.043841	0.061621	0.131485
11	0.005012	0.008881	0.009686	0.014576	0.021207	0.021621
	0.001539	0.004419	0.008734	0.018541	0.053077	0.060882
21	0.002023	0.002953	0.005251	0.016580	0.024810	0.044302
	0.001396	0.003481	0.009428	0.022292	0.057157	0.101220
101	0.001638	0.002910	0.005363	0.010694	0.023517	0.030562
	0.001223	0.005134	0.008143	0.017886	0.112510	0.273840
121	0.002725	0.005364	0.009598	0.016891	0.023798	0.149994
	0.002315	0.008874	0.019638	0.025271	0.143526	0.182596
131	0.001697	0.003414	0.006209	0.011061	0.020580	0.036597
	0.001750	0.006385	0.013237	0.023101	0.083554	0.083554
141	*0.274710*	*0.345206*	*0.368698*	*0.368698*	*0.368710*	*0.390440*
	0.000450	*0.003112*	*0.020969*	*0.180241*	*0.195543*	*N/A*[*]
151	*0.159593*	*0.270593*	*0.270593*	*0.283130*	*0.319881*	*0.544245*
	0.001587	*0.003423*	*0.009555*	*0.253358*	*0.264896*	*0.275277*
161	0.001210	0.003185	0.005423	0.016790	0.021774	0.039356
	0.002049	0.003349	0.006403	0.012046	0.032734	0.061718
171	0.001087	0.002799	0.005156	0.009667	0.015228	0.034612
	0.001078	0.002987	0.006094	0.014582	0.025857	0.053851
181	0.001622	0.004910	0.009234	0.027287	0.031706	0.031642
	0.001927	0.004567	0.007839	0.016862	0.061106	0.107411
191	*0.112850*	*0.369641*	*0.409859*	*0.409859*	*0.459248*	*0.461870*
	0.001463	*0.002904*	*0.007286*	*0.020929*	*0.027186*	*0.070660*
201	0.003040	0.011038	0.010554	0.017694	0.044833	0.051810
	0.002088	0.005312	0.012609	0.020542	0.045132	0.102804
211	0.001281	0.003021	0.014202	0.015640	0.015640	0.040179
	0.001638	0.003865	0.006155	0.012420	0.023633	0.041620
221	0.001176	0.003438	0.007301	0.008456	0.018090	0.045353
	0.001548	0.003648	0.009002	0.014075	0.037553	0.055626
231	0.002022	0.003345	0.011016	0.011394	0.021515	0.038959
	0.001671	0.004549	0.009484	0.017799	0.046244	0.054597

Table A-4: Hausdorff distances on 50% reduced meshes (cont'd)

Model	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
241	0.001499	0.004409	0.005709	0.010546	0.044435	0.040879
	0.001951	0.005069	0.010844	0.018118	0.069945	0.092267
251	0.001930	0.003842	0.006587	0.012511	0.036436	0.061209
	0.001977	0.006226	0.010290	0.027637	0.046993	0.046993
Average	0.032278	0.058908	0.065515	0.071308	0.088648	0.120839
	0.001799	0.005303	0.011457	0.042197	0.077126	0.105671
(no outliers)	0.002257	0.004994	0.008675	0.014790	0.029854	0.051903
	0.001925	0.005734	0.011227	0.020334	0.060043	0.096698
Max	0.274710	0.369641	0.409859	0.409859	0.459248	0.544245
	0.004732	0.018146	0.030507	0.253358	0.264896	0.275277
Min	0.001087	0.002799	0.005156	0.008456	0.015228	0.021621
	0.000450	0.002904	0.006094	0.012046	0.023633	0.041620

[*] N/A: The algorithm exhausted all possible contractions before the given level

Table A-5 shows results from a smaller experiment with an addition-based penalizing system, as described in Chapter V. The results from the original penalizing method are also shown for comparison. All of the models used in this experiment (except for the Canyon model) were fully manifold, therefore, β (weight for boundary penalty) was set to 0 for all models (for the Canyon model, β was set to 1). The α and δ weights (angular and regularity penalties) used in the experiment are displayed with the results.

Table A-5: Hausdorff results from addition-based (top) and logarithm-based (bottom) penalizing

Model	α	δ	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
1	.04	.04	0.003221	0.008186	0.020139	0.050261	0.046100	0.052993
			0.002224	0.005295	0.012277	0.032251	0.043783	0.055846
11	.04	.02	0.001773	0.005697	0.010630	0.013751	0.041870	0.059879
			0.001086	0.002134	0.004617	0.010499	0.042660	0.046877
21	.02	.02	0.000859	0.005629	0.012069	0.022804	0.041686	0.068408
			0.000828	0.001451	0.003182	0.005664	0.024215	0.036284
31	.02	.02	0.001046	0.004171	0.009114	0.013396	0.032604	0.044061
			0.000911	0.001629	0.002686	0.006126	0.016485	0.029743
41	.02	.01	0.003745	0.010782	0.014526	0.022632	0.044625	0.051597
			0.000735	0.002029	0.005805	0.012352	0.016966	0.029383
51	.04	.01	0.004701	0.009897	0.013608	0.021043	0.063311	0.063311
			0.001045	0.002857	0.007269	0.198070	0.035696	0.082072
61	.04	.02	0.001672	0.004870	0.007596	0.012780	0.024877	0.043107
			0.000936	0.002663	0.005099	0.007859	0.017481	0.048546
71	.04	.01	0.001347	0.003355	0.006685	0.008866	0.022761	0.039401
			0.000866	0.001814	0.004290	0.008492	0.015972	0.033649
81	.08	.01	0.003344	0.013319	0.023162	0.031626	0.082547	0.082547
			0.001463	0.004588	0.013379	0.022718	0.035793	0.065673
91	.04	.01	0.002704	0.010609	0.016245	0.026014	0.040292	0.060484
			0.001305	0.002815	0.006861	0.015804	0.039400	0.062954
101	.04	.01	0.002158	0.006011	0.013504	0.014754	0.033577	0.039902
			0.000358	0.001688	0.004069	0.007071	0.015602	0.024933
111	.04	.02	0.004063	0.008094	0.013912	0.020512	0.039066	0.069711
			0.001798	0.003412	0.009973	0.110990	0.137811	0.137811

Table A-5: Hausdorff results from addition-based (top) and logarithm-based (bottom) penalizing (cont’d)

Model	***α***	***δ***	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
121	.02	.01	0.003183	0.013139	0.023718	0.029295	0.038815	0.104132
			0.001723	0.003308	0.009335	0.016785	0.035186	0.054382
131	.04	.01	0.003925	0.007276	0.014393	0.024043	0.045249	0.063384
			0.001141	0.002175	0.004784	0.009780	0.039497	0.039497
141	.04	.01	0.000007	0.001240	0.004102	0.010815	0.021084	0.059335
			0.000000	0.000798	0.002153	0.004322	0.018364	0.021316
151	.04	.01	0.000027	0.003362	0.005408	0.008219	0.013736	0.024338
			0.000002	0.001568	0.002214	0.004966	0.009630	0.019507
161	.02	.01	0.001093	0.004736	0.012361	0.022681	0.028093	0.036708
			0.001057	0.002250	0.003568	0.006740	0.018576	0.030684
171	.02	.01	0.001049	0.004368	0.009773	0.017931	0.030683	0.057850
			0.000819	0.001342	0.002617	0.005998	0.018948	0.026481
181	.02	.02	0.001498	0.006862	0.009167	0.019401	0.042370	0.044638
			0.001321	0.002688	0.004843	0.009168	0.026919	0.033829
191	.02	.01	0.001277	0.003562	0.010515	0.012513	0.026983	0.054035
			0.000960	0.002136	0.004479	0.005807	0.023554	0.028798
201	.02	.02	0.001618	0.004905	0.006739	0.011644	0.023292	0.031477
			0.001128	0.002959	0.005412	0.011096	0.017711	0.027170
211	.04	.04	0.001473	0.004357	0.008335	0.020163	0.027414	0.031591
			0.001424	0.002741	0.006083	0.009328	0.021282	0.028368
221	.04	.02	0.001003	0.005319	0.011424	0.016682	0.028560	0.054341
			0.000961	0.002068	0.004672	0.008029	0.023256	0.028294
231	.04	.01	0.001171	0.007742	0.012605	0.022419	0.038736	0.051755
			0.000964	0.002802	0.004548	0.007698	0.021810	0.040588
241	.04	.01	0.001866	0.006512	0.008518	0.013869	0.020003	0.081559
			0.000936	0.002048	0.005365	0.010416	0.021014	0.030753
251	.04	.01	0.002925	0.006509	0.013227	0.014528	0.026519	0.039704
			0.001559	0.002748	0.005973	0.012124	0.030985	0.057274
281	.02	.02	0.001511	0.004942	0.009003	0.020984	0.035004	0.050837
			0.001501	0.002587	0.004435	0.006124	0.020837	0.035884
291	.02	.01	0.001852	0.006249	0.015005	0.022746	0.048905	0.047393
			0.001786	0.003055	0.004644	0.007911	0.019760	0.037161
301	.06	.01	0.001672	0.005768	0.011444	0.024936	0.053522	0.061029
			0.001613	0.003367	0.004583	0.007732	0.021307	0.045735
311	.04	.01	0.001423	0.004729	0.011786	0.018082	0.024727	0.031528
			0.001211	0.002245	0.003470	0.005399	0.010804	0.020043
313	.04	.01	0.007135	0.007461	0.014864	0.027092	0.038751	0.065515
			0.002407	0.004258	0.007805	0.012166	0.026915	0.048070

Table A-5: Hausdorff results from addition-based penalizing (cont'd)

Model	***α***	***δ***	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
321	.04	.02	0.000000	0.000569	0.003871	0.008705	0.012630	0.015814
			0.000000	0.000000	0.000005	0.000006	0.025466	0.120618
331	.04	.01	0.000000	0.000007	0.000007	0.011459	0.011087	0.027787
			0.000000	0.000000	0.000005	0.000007	0.020965	0.076975
341	.04	.01	0.029733	0.141720	0.162863	0.211988	0.220224	0.220224
			0.025949	0.115654	0.195246	0.197037	0.197037	0.197037
351	.02	.01	0.001526	0.019115	0.088408	N/A[*]	N/A[*]	N/A[*]
			0.006076	0.001245	0.084882	N/A[*]	N/A[*]	N/A[*]
361	.06	.01	0.002499	0.010469	0.016088	0.022784	0.032267	0.067565
			0.001337	0.002258	0.005189	0.011103	0.038002	0.081720
371	.04	.01	0.000696	0.002157	0.004087	0.008030	0.019108	0.029171
			0.000681	0.001358	0.002228	0.003419	0.007498	0.017142
381	.02	.01	0.002091	0.010843	0.023459	0.045668	0.058461	0.053997
			0.001673	0.005275	0.007882	0.017825	0.055342	0.068584
391	.04	.01	0.008792	0.016667	0.024697	0.037599	0.065090	0.065454
			0.003351	0.006442	0.013083	0.031258	0.051090	0.088611
Canyon	.02	.02	0.001684	0.006566	0.011067	0.022801	0.030387	0.035161
	β: 1		0.004899	0.012800	0.028362	0.033196	0.033319	0.034957
Horse	.04	.01	0.000566	0.002303	0.005272	0.008983	0.014943	0.025378
			0.000490	0.001019	0.004283	0.004283	0.005694	0.014777

[*] N/A: The algorithm exhausted all possible contractions before the given level

Figures A-1 to A-13 show graphs of the Hausdorff distances between our method and the QEM method for selected models.

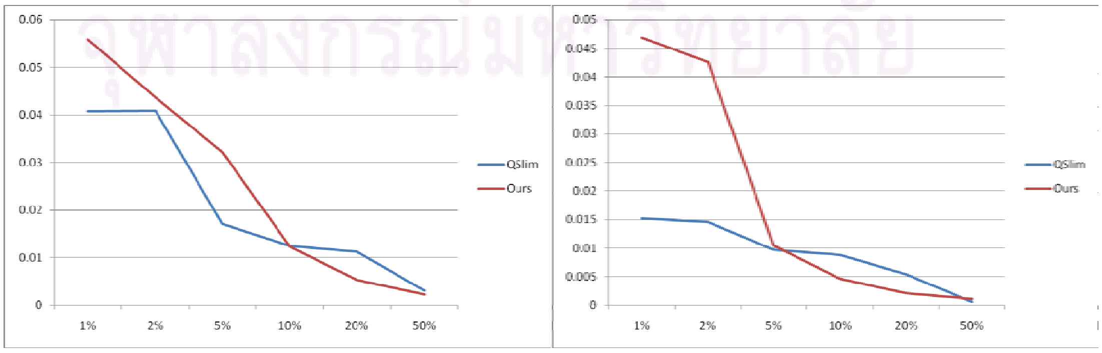


Figure A-1: Hausdorff distances for Female (left) and Male (right) models

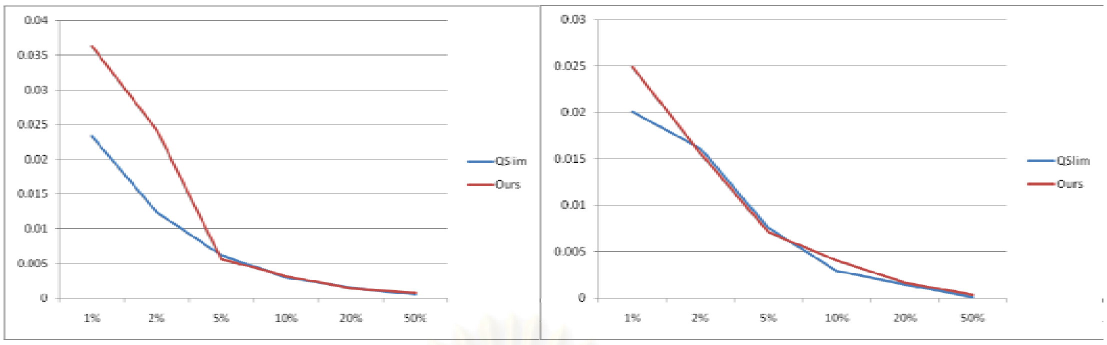


Figure A-2: Hausdorff distances for Cup (left) and Chair (right) models

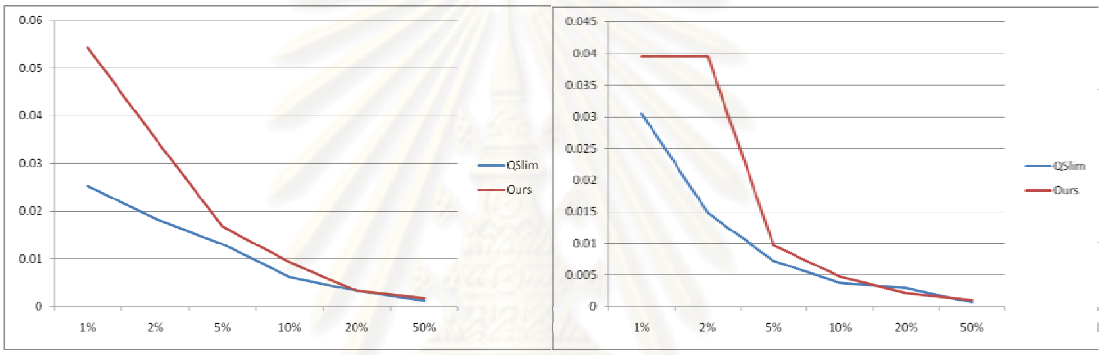


Figure A-3: Hausdorff distances for Squid (left) and Squid 2 (right) models

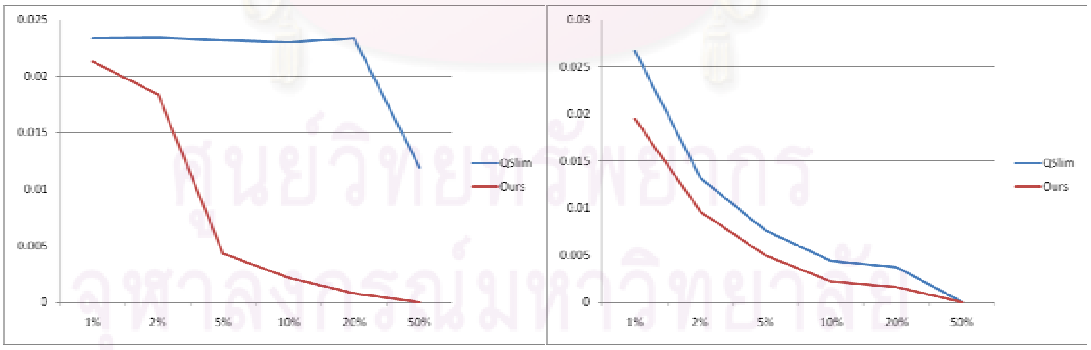


Figure A-4: Hausdorff distances for Table (left) and Table 2 (right) models

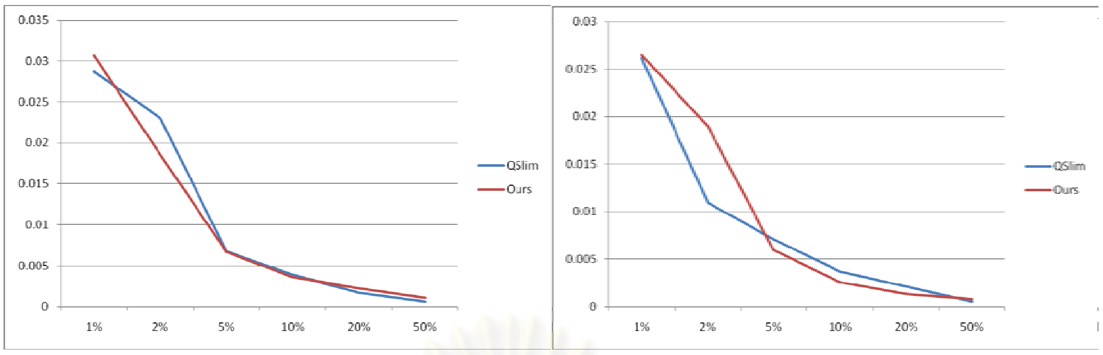


Figure A-5: Hausdorff distances for Teddy (left) and Teddy 2 (right) models

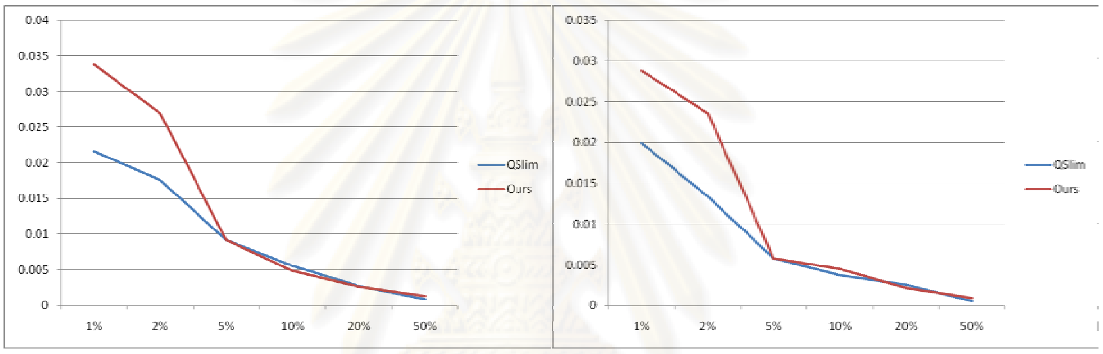


Figure A-6: Hausdorff distances for Hand (left) and Hand 2 (right) models

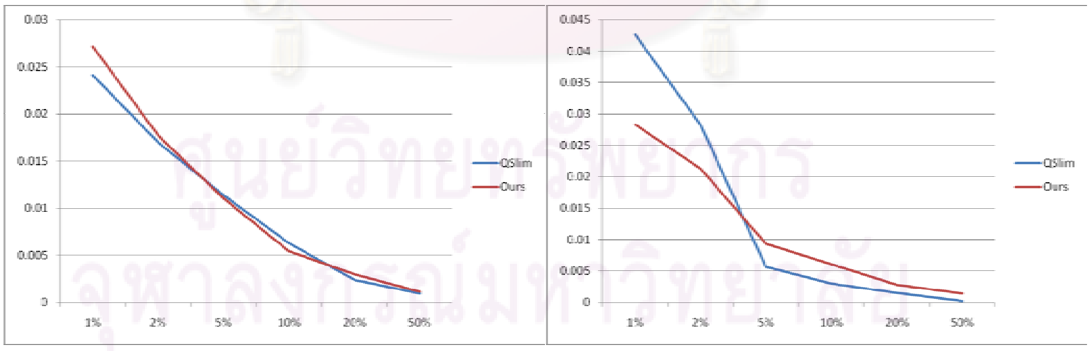


Figure A-7: Hausdorff distances for Pliers (left) and Pliers 2 (right) models

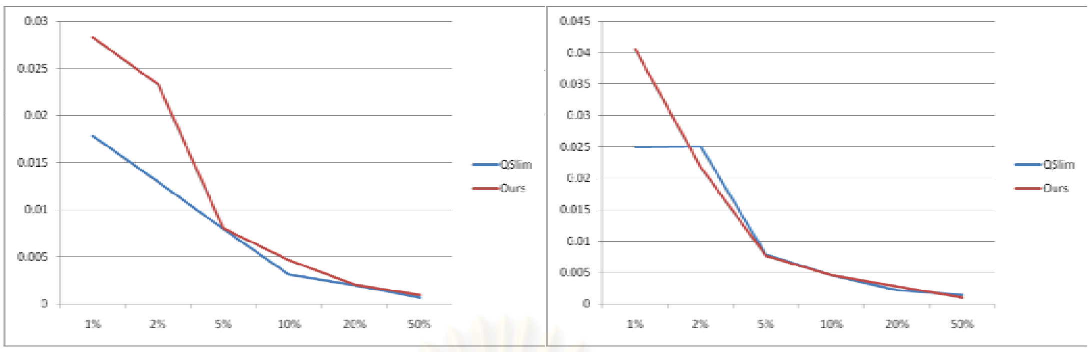


Figure A-8: Hausdorff distances for Dolphin (left) and Fish (right) models

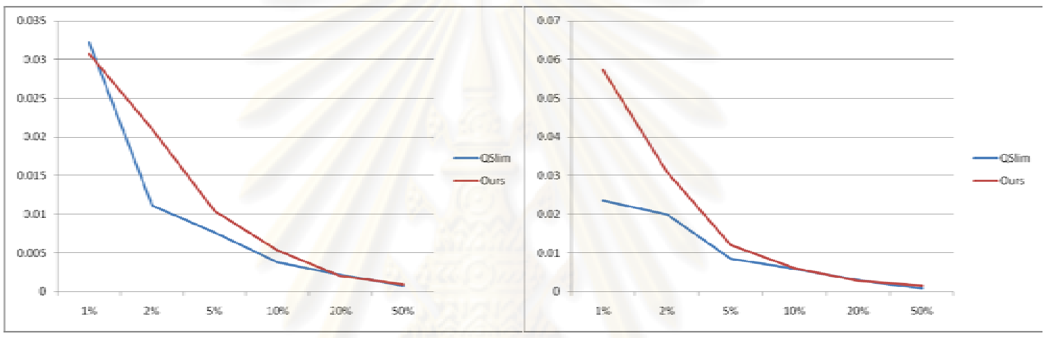


Figure A-9: Hausdorff distances for Bird (left) and Bird 2 (right) models

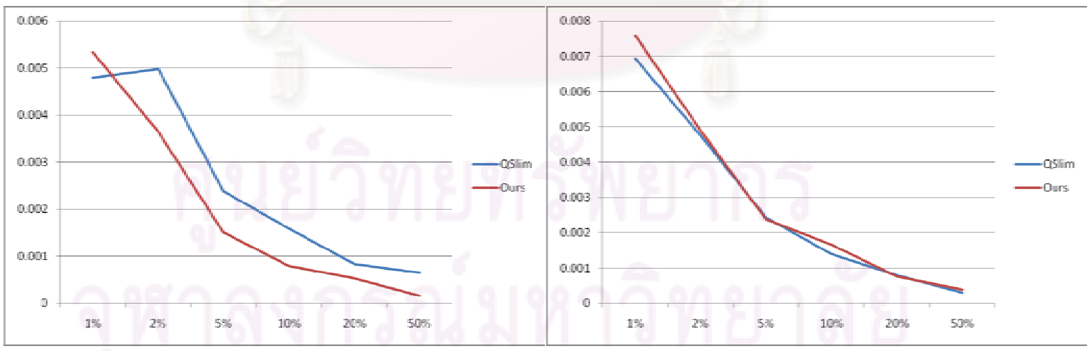


Figure A-10: Hausdorff distances for Angel (left) and Armadillo (right) models

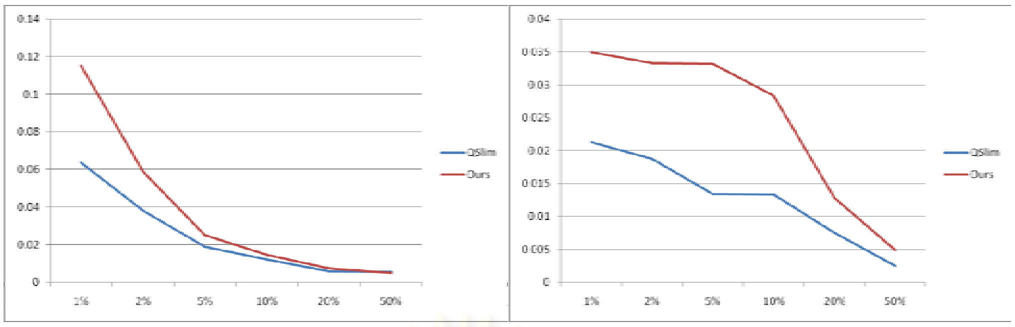


Figure A-11: Hausdorff distances for Bunny (left) and Canyon (right) models

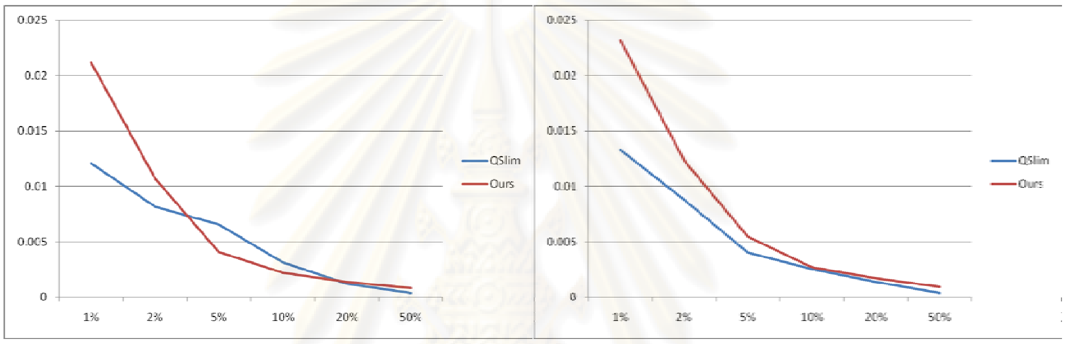


Figure A-12: Hausdorff distances for Dinosaur (left) and Dragon (right) models

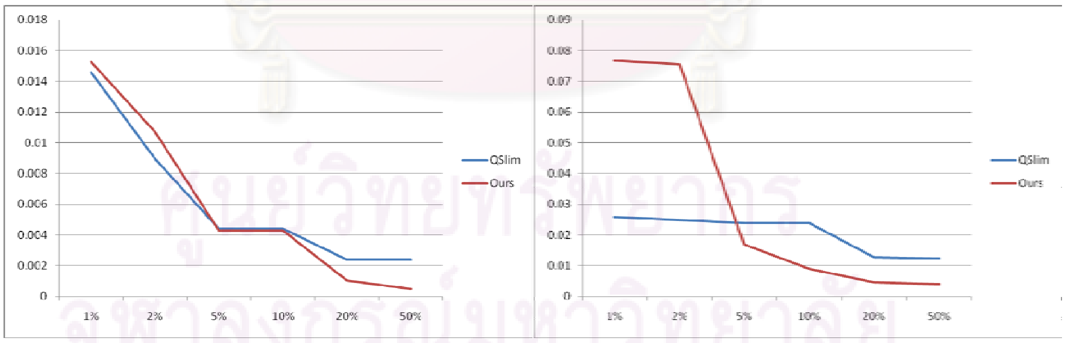


Figure A-13: Hausdorff distances for Horse (left) and Turbine (right) models

A.2 Running times with and without partial updates, and with different updating parameters

In Table A-6, we compare the running times of selected models when running with (top row) and without (middle row) partial updates, and display the ratios of the running times (bottom row).

Table A-6: Comparing running time with and without partial updates: (top row) with partial updates, (middle row) without partial updates, (bottom row) ratio

Model	50%	20%	10%	5%	2%	1%
1	19.158	33.048	35.902	36.953	38.445	39.006
	114.719	174.953	191.328	198.203	201.844	202.891
	5.988	*5.294*	*5.329*	*5.364*	*5.250*	*5.202*
11	50.633	78.583	86.554	89.479	91.271	91.862
	254.563	375.734	414.578	432.313	440.563	443.047
	5.028	*4.781*	*4.790*	*4.831*	*4.827*	*4.823*
21	52.185	78.583	86.554	89.479	91.271	91.862
	375.688	550.969	602.013	623.469	634.875	638.313
	7.199	*7.011*	*6.955*	*6.968*	*6.956*	*6.949*
31	54.919	92.553	104.100	108.075	111.310	112.832
	373.641	543.172	590.781	611.391	621.906	625.406
	6.803	*5.869*	*5.675*	*5.657*	*5.587*	*5.543*
61	19.418	33.078	36.202	38.355	39.817	40.338
	116.938	170.391	184.984	191.266	194.844	196.016
	6.022	*5.151*	*5.110*	*4.987*	*4.893*	*4.859*
81	21.721	37.143	41.029	43.392	45.045	45.816
	138.234	203.859	224.969	233.516	238.234	239.469
	6.364	*5.488*	*5.483*	*5.382*	*5.289*	*5.227*
111	38.344	62.891	70.203	71.500	73.047	73.219
	339.438	430.922	456.250	469.594	476.734	479.016
	8.852	*6.852*	*6.499*	*6.568*	*6.526*	*6.542*
121	23.484	38.896	42.571	45.065	47.488	48.700
	148.516	211.203	227.156	234.125	239.047	240.047
	6.324	*5.430*	*5.336*	*5.195*	*5.034*	*4.929*
131	48.547	81.813	92.328	95.781	98.172	99.031
	276.891	399.906	435.188	448.875	456.859	459.031
	5.704	*4.888*	*4.713*	*4.686*	*4.654*	*4.635*
141	62.859	98.843	111.344	116.484	119.813	120.813
	634.078	844.766	896.344	920.625	931.531	934.422
	10.087	*8.547*	*8.050*	*7.903*	*7.775*	*7.734*

Table A-6: Comparing running time with and without partial updates: (top row) with partial updates, (middle row) without partial updates, (bottom row) ratio (cont'd)

Model	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
151	48.688	82.844	95.750	100.156	103.156	103.875
	494.343	665.609	706.828	723.828	733.016	735.734
	10.153	*8.034*	*7.382*	*7.227*	*7.106*	*7.083*
171	52.505	89.549	100.525	104.630	107.845	109.087
	357.141	520.406	564.344	585.484	596.813	599.891
	6.802	*5.811*	*5.614*	*5.596*	*5.534*	*5.499*
181	27.844	47.328	51.391	54.047	56.141	57.141
	155.203	230.906	250.859	262.500	267.953	269.594
	5.574	*4.879*	*4.881*	*4.857*	*4.773*	*4.718*
191	56.722	96.819	111.000	114.995	119.061	120.223
	337.734	509.531	556.938	577.281	587.172	590.343
	5.954	*5.263*	*5.017*	*5.020*	*4.932*	*4.910*
201	18.136	27.520	30.604	32.657	34.039	34.710
	103.734	154.343	168.984	174.969	178.109	178.969
	5.720	*5.608*	*5.522*	*5.358*	*5.232*	*5.156*
211	17.797	27.984	31.156	32.891	34.031	34.641
	98.906	145.797	162.891	168.547	171.844	172.844
	5.557	*5.210*	*5.228*	*5.124*	*5.050*	*4.990*
221	22.873	39.437	43.713	46.186	47.468	48.129
	165.250	241.906	262.125	271.078	275.906	277.297
	7.225	*6.134*	*5.996*	*5.869*	*5.812*	*5.762*
231	24.688	42.156	46.625	49.109	50.391	51.109
	142.688	204.609	223.203	230.813	234.984	236.359
	5.780	*4.854*	*4.787*	*4.700*	*4.663*	*4.625*
241	14.844	22.578	24.984	26.141	27.328	27.891
	69.438	102.719	111.672	116.000	118.125	118.859
	4.678	*4.550*	*4.470*	*4.437*	*4.322*	*4.262*
251	12.829	21.391	23.813	24.969	25.922	26.297
	65.438	96.313	104.359	107.859	110.297	110.953
	5.101	*4.503*	*4.382*	*4.320*	*4.255*	*4.219*
281	105.682	178.196	197.905	207.599	211.694	213.697
	664.609	969.344	1047.641	1086.172	1108.375	1114.891
	6.289	*5.440*	*5.294*	*5.232*	*5.236*	*5.217*
291	86.855	144.988	162.183	171.186	174.421	176.033
	622.922	923.109	1008.250	1042.344	1061.484	1067.453
	7.172	*6.367*	*6.217*	*6.089*	*6.086*	*6.064*
301	33.328	56.371	61.378	64.473	66.235	66.866
	203.984	301.516	330.844	348.469	355.891	357.984
	6.120	*5.349*	*5.390*	*5.405*	*5.373*	*5.354*
311	100.104	171.947	193.108	203.703	208.300	209.942
	844.359	1183.766	1266.609	1302.000	1320.469	1326.438
	8.435	*6.884*	*6.559*	*6.392*	*6.339*	*6.318*

Table A-6: Comparing running time with and without partial updates: (top row) with partial updates, (middle row) without partial updates, (bottom row) ratio (cont'd)

Model	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
321	81.838	123.698	134.443	138.860	141.503	142.204
	427.766	624.266	686.578	718.297	733.031	738.453
	5.227	*5.047*	*5.107*	*5.173*	*5.180*	*5.193*
331	77.031	128.938	143.719	148.719	151.781	153.203
	435.813	655.078	722.594	751.891	766.891	771.094
	5.658	*5.081*	*5.028*	*5.056*	*5.053*	*5.033*
341	4.777	7.751	8.512	8.943	9.303	9.544
	32.109	55.328	60.047	62.500	63.781	64.141
	6.722	*7.138*	*7.054*	*6.989*	*6.856*	*6.721*
351	18.987	35.030	36.693	N/A[*]	N/A[*]	N/A[*]
	125.359	175.141	190.109	N/A[*]	N/A[*]	N/A[*]
	6.602	*5.000*	*5.181*	N/A[*]	N/A[*]	N/A[*]
361	49.631	88.770	93.965	97.891	100.955	102.267
	312.938	467.938	510.234	529.391	539.813	543.531
	6.305	*5.271*	*5.430*	*5.408*	*5.347*	*5.315*
371	58.394	87.265	98.652	104.530	107.044	108.426
	306.875	452.984	491.719	509.453	519.391	522.531
	5.255	*5.191*	*4.984*	*4.874*	*4.852*	*4.819*
381	28.150	46.206	50.483	52.846	54.398	55.430
	134.938	200.953	220.609	229.438	234.406	235.766
	4.794	*4.349*	*4.370*	*4.342*	*4.309*	*4.253*
391	14.551	22.643	25.266	26.448	27.810	28.571
	81.016	121.563	132.594	137.547	140.156	140.875
	5.568	*5.369*	*5.248*	*5.201*	*5.040*	*4.931*
Avg.	***6.408***	***5.645***	***5.534***	***5.491***	***5.424***	***5.383***

[*] N/A: The algorithm exhausted all possible contractions before the given level

Table A-7 displays the running times when changing the parameters used in our updating scheme as described in Chapter V, along with running times when not using caching. For comparison, we also display the running times using normal parameters.

Table A-7: Running times after changing update parameters

Model	**Parameters**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
1	*n*-8 layers	25.891	41.422	47.578	51.422	53.828	54.859
	n-4 layers	28.703	43.906	47.641	49.484	50.063	50.266
	$\lvert C\rvert \geq \lvert H\rvert \times 6$	13.797	25.656	30.328	32.391	33.844	34.516
	$\lvert C\rvert \geq \lvert H\rvert \times 2$	33.969	49.861	52.984	54.234	55.500	56.234
	No Cache	19.438	35.703	38.422	39.484	40.844	41.422
	Regular	19.158	33.048	35.902	36.953	38.445	39.006
11	*n*-8 layers	51.359	84.016	99.688	105.531	108.922	112.578
	n-4 layers	62.516	95.469	109.156	112.656	114.547	114.984
	$\lvert C\rvert \geq \lvert H\rvert \times 6$	30.594	57.672	69.031	73.125	75.703	76.703
	$\lvert C\rvert \geq \lvert H\rvert \times 2$	113.188	157.672	171.078	176.547	179.578	180.672
	No Cache	48.844	77.969	86.828	90.078	92.219	93.016
	Regular	50.633	78.583	86.554	89.479	91.271	91.862
21	*n*-8 layers	67.797	102.078	116.938	124.813	127.938	130.094
	n-4 layers	85.453	128.547	145.984	152.172	154.547	155.172
	$\lvert C\rvert \geq \lvert H\rvert \times 6$	43.953	80.719	94.797	99.016	101.844	102.859
	$\lvert C\rvert \geq \lvert H\rvert \times 2$	78.594	140.031	152.219	159.109	161.734	162.656
	No Cache	60.500	103.156	115.375	119.750	123.094	124.203
	Regular	52.185	78.583	86.554	89.479	91.271	91.862
31	*n*-8 layers	81.672	92.859	107.547	115.375	118.453	121.328
	n-4 layers	84.328	128.891	146.703	153.391	156.250	157.109
	$\lvert C\rvert \geq \lvert H\rvert \times 6$	47.250	87.234	102.188	106.844	110.125	111.484
	$\lvert C\rvert \geq \lvert H\rvert \times 2$	77.203	139.891	152.531	159.563	162.375	163.484
	No Cache	62.547	106.469	121.672	126.797	130.797	132.344
	Regular	54.919	92.553	104.100	108.075	111.310	112.832
41	*n*-8 layers	25.672	40.594	47.750	50.500	52.969	54.297
	n-4 layers	41.141	66.328	70.703	73.344	74.828	75.250
	$\lvert C\rvert \geq \lvert H\rvert \times 6$	21.719	38.813	45.750	48.266	49.531	50.453
	$\lvert C\rvert \geq \lvert H\rvert \times 2$	39.016	58.906	66.016	68.578	70.031	70.938
	No Cache	29.875	50.031	54.391	57.078	58.594	59.547
	Regular	23.023	38.936	42.431	44.474	45.616	46.347

Table A-7: Running times after changing update parameters (cont'd)

Model	**Parameters**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
51	*n*-8 layers	11.344	17.797	19.766	22.109	23.734	24.172
	n-4 layers	18.250	25.875	28.094	28.953	29.688	29.875
	$\|C\|\geq\|H\|\times 6$	9.219	16.578	18.750	19.953	21.250	21.984
	$\|C\|\geq\|H\|\times 2$	13.781	20.250	21.906	23.438	24.578	25.359
	No Cache	11.594	18.391	20.703	21.781	22.813	23.563
	Regular	9.664	15.262	17.145	18.326	19.158	19.708
61	*n*-8 layers	20.422	33.094	38.656	40.688	43.516	44.547
	n-4 layers	30.422	46.484	51.703	52.938	54.063	54.313
	$\|C\|\geq\|H\|\times 6$	17.625	31.234	36.703	38.188	39.547	40.250
	$\|C\|\geq\|H\|\times 2$	31.141	46.875	50.797	52.953	54.359	55.063
	No Cache	22.641	39.000	42.531	44.813	46.328	46.813
	Regular	19.418	33.078	36.202	38.355	39.817	40.338
71	*n*-8 layers	21.328	34.141	39.953	42.063	44.531	45.734
	n-4 layers	31.938	52.516	56.281	58.484	59.438	59.953
	$\|C\|\geq\|H\|\times 6$	17.969	32.422	38.297	39.906	41.563	42.031
	$\|C\|\geq\|H\|\times 2$	32.078	48.078	52.375	54.797	56.203	56.781
	No Cache	25.656	42.125	45.813	48.031	49.484	50.141
	Regular	19.598	32.597	35.461	37.224	38.295	38.776
81	*n*-8 layers	24.234	38.594	45.188	48.234	51.906	53.156
	n-4 layers	36.688	55.906	63.031	65.688	67.203	67.516
	$\|C\|\geq\|H\|\times 6$	19.844	35.875	42.313	44.578	46.203	47.063
	$\|C\|\geq\|H\|\times 2$	37.359	54.922	61.047	62.906	65.109	65.922
	No Cache	27.516	46.656	50.984	54.203	56.375	58.094
	Regular	21.721	37.143	41.029	43.392	45.045	45.816
91	*n*-8 layers	32.672	51.297	60.234	64.578	68.422	69.953
	n-4 layers	49.375	75.563	86.250	89.234	90.578	91.359
	$\|C\|\geq\|H\|\times 6$	26.281	49.266	57.844	61.094	63.453	64.172
	$\|C\|\geq\|H\|\times 2$	49.594	77.766	82.844	86.328	88.625	89.734
	No Cache	43.578	75.203	83.156	85.891	87.422	88.625
	Regular	28.511	48.299	54.298	56.421	57.553	58.414
101	*n*-8 layers	33.203	62.359	76.234	81.859	85.984	88.188
	n-4 layers	48.344	74.594	85.672	88.578	90.016	90.797
	$\|C\|\geq\|H\|\times 6$	30.438	48.906	57.016	60.047	62.016	62.781
	$\|C\|\geq\|H\|\times 2$	47.781	83.906	99.469	104.375	106.656	108.469
	No Cache	43.672	70.375	79.094	82.125	83.984	85.313
	Regular	35.891	59.250	66.953	70.047	71.172	71.781
111	*n*-8 layers	39.250	65.016	76.219	80.703	83.313	N/A[*]
	n-4 layers	52.547	78.500	88.844	91.703	92.906	93.094
	$\|C\|\geq\|H\|\times 6$	27.578	47.625	55.984	59.000	60.281	60.391
	$\|C\|\geq\|H\|\times 2$	66.031	101.250	108.625	111.484	112.734	112.750
	No Cache	39.141	63.891	73.656	76.281	78.766	79.078
	Regular	38.344	62.891	70.203	71.500	73.047	73.219

Table A-7: Running times after changing update parameters (cont’d)

Model	Parameters	50%	20%	10%	5%	2%	1%
121	*n*-8 layers	27.578	43.594	51.422	54.625	58.000	59.813
	n-4 layers	34.000	50.563	55.688	56.922	58.047	58.344
	$\|C\|\geq\|H\|\times 6$	20.500	35.297	40.953	42.719	44.625	45.297
	$\|C\|\geq\|H\|\times 2$	29.328	51.688	55.609	57.875	59.344	60.281
	No Cache	25.234	42.766	46.484	49.063	50.953	51.844
	Regular	23.484	38.896	42.571	45.065	47.488	48.700
131	*n*-8 layers	65.563	99.563	114.031	119.469	123.750	126.516
	n-4 layers	65.422	95.984	109.344	112.766	114.750	115.250
	$\|C\|\geq\|H\|\times 6$	37.359	64.313	75.031	79.203	82.078	83.703
	$\|C\|\geq\|H\|\times 2$	47.359	93.813	104.031	108.297	111.234	112.734
	No Cache	45.328	75.828	85.109	88.641	91.234	92.328
	Regular	48.547	81.813	92.328	95.781	98.172	99.031
141	*n*-8 layers	59.516	93.531	105.391	110.406	114.109	115.781
	n-4 layers	66.375	107.531	123.750	129.938	132.203	132.641
	$\|C\|\geq\|H\|\times 6$	68.344	100.688	113.672	118.688	122.094	123.438
	$\|C\|\geq\|H\|\times 2$	80.141	139.453	157.328	165.531	169.891	170.688
	No Cache	61.125	95.438	107.656	112.656	116.141	117.219
	Regular	62.859	98.843	111.344	116.484	119.813	120.813
151	*n*-8 layers	41.797	80.500	91.266	95.313	99.250	101.938
	n-4 layers	56.094	97.281	108.219	114.281	116.703	117.219
	$\|C\|\geq\|H\|\times 6$	50.531	75.031	90.109	94.813	98.031	99.234
	$\|C\|\geq\|H\|\times 2$	63.109	105.109	120.797	125.063	127.922	128.969
	No Cache	49.828	81.703	93.375	97.813	100.688	101.531
	Regular	48.688	82.844	95.750	100.156	103.156	103.875
161	*n*-8 layers	59.484	93.281	108.391	117.703	121.000	122.953
	n-4 layers	80.500	129.281	141.625	147.953	150.438	151.078
	$\|C\|\geq\|H\|\times 6$	41.797	75.391	88.250	92.922	96.328	97.609
	$\|C\|\geq\|H\|\times 2$	82.578	128.844	144.813	149.063	152.359	153.813
	No Cache	57.531	97.297	108.938	113.188	116.438	117.828
	Regular	66.976	102.497	112.121	115.586	118.210	119.452
171	*n*-8 layers	67.156	104.641	120.547	128.625	131.969	133.594
	n-4 layers	85.938	137.859	150.547	157.625	160.516	161.516
	$\|C\|\geq\|H\|\times 6$	44.453	81.188	95.547	102.531	104.922	105.734
	$\|C\|\geq\|H\|\times 2$	88.563	137.703	154.906	159.578	163.234	164.719
	No Cache	64.063	108.172	122.125	127.156	131.219	132.766
	Regular	41.329	70.541	79.094	82.308	84.572	85.883
181	*n*-8 layers	28.859	46.156	53.672	56.438	59.453	61.125
	n-4 layers	41.063	62.094	68.750	71.125	72.453	72.703
	$\|C\|\geq\|H\|\times 6$	19.750	36.297	42.750	44.859	45.984	46.719
	$\|C\|\geq\|H\|\times 2$	35.438	62.094	68.531	70.234	72.391	73.344
	No Cache	29.547	49.266	53.375	56.156	57.750	58.531
	Regular	27.844	47.328	51.391	54.047	56.141	57.141

Table A-7: Running times after changing update parameters (cont’d)

Model	**Parameters**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
191	*n*-8 layers	54.047	84.766	99.031	106.922	112.172	114.531
	n-4 layers	79.453	119.875	136.250	142.266	144.344	144.984
	$\|C\|\geq\|H\|\times 6$	40.969	76.250	91.391	95.844	99.031	100.453
	$\|C\|\geq\|H\|\times 2$	81.328	125.141	140.594	144.359	147.172	148.141
	No Cache	55.078	93.859	105.625	109.734	112.750	113.813
	Regular	56.722	96.819	111.000	114.995	119.061	120.223
201	*n*-8 layers	18.766	29.516	33.641	36.641	39.422	40.391
	n-4 layers	23.250	41.938	44.875	45.813	46.828	47.047
	$\|C\|\geq\|H\|\times 6$	13.406	23.969	27.078	28.984	29.922	30.344
	$\|C\|\geq\|H\|\times 2$	22.625	36.438	39.688	41.344	42.156	42.813
	No Cache	19.203	29.625	32.891	34.844	36.000	36.547
	Regular	18.136	27.520	30.604	32.657	34.039	34.710
211	*n*-8 layers	21.031	32.578	37.469	39.813	42.641	43.500
	n-4 layers	25.375	37.641	40.250	41.156	41.984	42.219
	$\|C\|\geq\|H\|\times 6$	15.141	27.922	31.500	33.438	34.594	35.188
	$\|C\|\geq\|H\|\times 2$	21.969	35.516	38.469	39.609	40.828	41.656
	No Cache	18.000	28.313	31.547	33.328	34.516	35.141
	Regular	17.797	27.984	31.156	32.891	34.031	34.641
221	*n*-8 layers	31.625	48.109	56.313	59.688	62.469	64.281
	n-4 layers	39.969	62.859	69.766	72.266	73.531	73.859
	$\|C\|\geq\|H\|\times 6$	21.469	40.750	47.781	50.094	51.656	52.453
	$\|C\|\geq\|H\|\times 2$	34.234	60.375	66.891	69.172	70.375	71.141
	No Cache	28.641	49.031	53.484	56.438	58.094	58.953
	Regular	22.873	39.437	43.713	46.186	47.468	48.129
231	*n*-8 layers	24.094	40.609	47.766	50.922	54.500	56.016
	n-4 layers	34.250	52.938	59.109	61.266	62.234	62.641
	$\|C\|\geq\|H\|\times 6$	19.656	35.203	41.797	43.813	44.813	45.234
	$\|C\|\geq\|H\|\times 2$	35.734	52.813	58.563	60.047	61.578	62.250
	No Cache	26.031	44.328	48.609	51.203	52.453	53.156
	Regular	24.688	42.156	46.625	49.109	50.391	51.109
241	*n*-8 layers	15.359	24.484	28.469	31.219	33.875	34.781
	n-4 layers	16.109	27.406	29.875	31.063	31.484	31.672
	$\|C\|\geq\|H\|\times 6$	10.344	18.906	21.422	22.516	23.438	23.781
	$\|C\|\geq\|H\|\times 2$	17.906	26.438	29.219	30.578	32.219	32.906
	No Cache	14.531	22.531	25.094	26.328	27.578	28.172
	Regular	14.844	22.578	24.984	26.141	27.328	27.891
251	*n*-8 layers	15.844	23.516	26.953	30.563	32.781	33.375
	n-4 layers	16.859	29.500	32.172	33.516	33.891	34.031
	$\|C\|\geq\|H\|\times 6$	9.625	17.781	20.375	21.703	23.156	23.922
	$\|C\|\geq\|H\|\times 2$	15.359	22.688	25.188	26.406	27.578	28.109
	No Cache	13.047	20.563	23.000	24.172	25.203	25.672
	Regular	12.829	21.391	23.813	24.969	25.922	26.297

Table A-7: Running times after changing update parameters (cont'd)

Model	**Parameters**	**50%**	**20%**	**10%**	**5%**	**2%**	**1%**
281	*n*-8 layers	114.859	179.906	208.578	222.641	229.984	231.813
	n-4 layers	153.328	229.391	259.344	270.516	277.031	278.578
	$\|C\|\geq\|H\|\times6$	88.250	161.719	199.078	207.094	213.875	215.328
	$\|C\|\geq\|H\|\times2$	158.047	238.438	266.266	277.313	283.344	284.719
	No Cache	106.797	177.281	197.484	207.625	211.750	213.625
	Regular	105.682	178.196	197.905	207.599	211.694	213.697
291	*n*-8 layers	109.438	164.156	188.438	200.250	206.453	208.063
	n-4 layers	148.281	224.109	253.891	265.188	271.484	273.000
	$\|C\|\geq\|H\|\times6$	84.250	150.047	173.922	182.063	189.047	190.891
	$\|C\|\geq\|H\|\times2$	154.594	233.266	260.563	271.391	277.188	278.422
	No Cache	104.891	174.172	194.000	203.969	207.969	210.000
	Regular	86.855	144.988	162.183	171.186	174.421	176.033
301	*n*-8 layers	36.547	58.000	67.594	71.453	74.922	77.313
	n-4 layers	52.859	84.984	92.891	95.938	97.813	98.234
	$\|C\|\geq\|H\|\times6$	30.359	58.438	63.484	66.578	68.688	69.516
	$\|C\|\geq\|H\|\times2$	53.422	85.016	93.250	96.109	97.813	98.797
	No Cache	37.375	65.109	73.875	78.547	80.156	81.672
	Regular	33.328	56.371	61.378	64.473	66.235	66.866
311	*n*-8 layers	100.453	171.516	185.891	198.688	205.578	207.094
	n-4 layers	168.156	279.563	307.063	325.984	332.203	333.344
	$\|C\|\geq\|H\|\times6$	132.859	190.219	205.406	220.281	225.156	227.094
	$\|C\|\geq\|H\|\times2$	131.609	235.516	263.828	275.000	280.906	281.922
	No Cache	116.594	194.344	218.688	230.219	234.828	236.766
	Regular	100.104	171.947	193.108	203.703	208.300	209.942
321	*n*-8 layers	74.438	122.234	131.453	140.750	143.625	144.188
	n-4 layers	128.844	179.438	193.281	202.219	206.500	207.703
	$\|C\|\geq\|H\|\times6$	112.375	170.641	181.203	187.922	192.500	193.438
	$\|C\|\geq\|H\|\times2$	103.453	152.719	165.828	174.156	177.156	177.844
	No Cache	101.625	150.297	163.656	168.750	172.047	172.922
	Regular	81.838	123.698	134.443	138.860	141.503	142.204
331	*n*-8 layers	58.359	103.734	113.000	120.281	122.891	125.156
	n-4 layers	111.172	160.797	178.141	187.938	192.313	193.953
	$\|C\|\geq\|H\|\times6$	84.047	140.219	151.563	158.156	161.922	162.594
	$\|C\|\geq\|H\|\times2$	95.609	146.953	164.391	169.531	172.938	173.391
	No Cache	105.969	172.641	191.984	198.656	203.063	203.875
	Regular	77.031	128.938	143.719	148.719	151.781	153.203
341	*n*-8 layers	4.578	7.594	8.609	10.609	11.594	11.953
	n-4 layers	9.766	14.516	15.859	16.453	16.953	17.250
	$\|C\|\geq\|H\|\times6$	5.766	8.656	9.469	9.922	10.156	10.266
	$\|C\|\geq\|H\|\times2$	5.156	8.438	9.234	9.641	9.984	10.250
	No Cache	6.813	11.063	12.078	12.578	12.984	13.297
	Regular	4.777	7.751	8.512	8.943	9.303	9.544

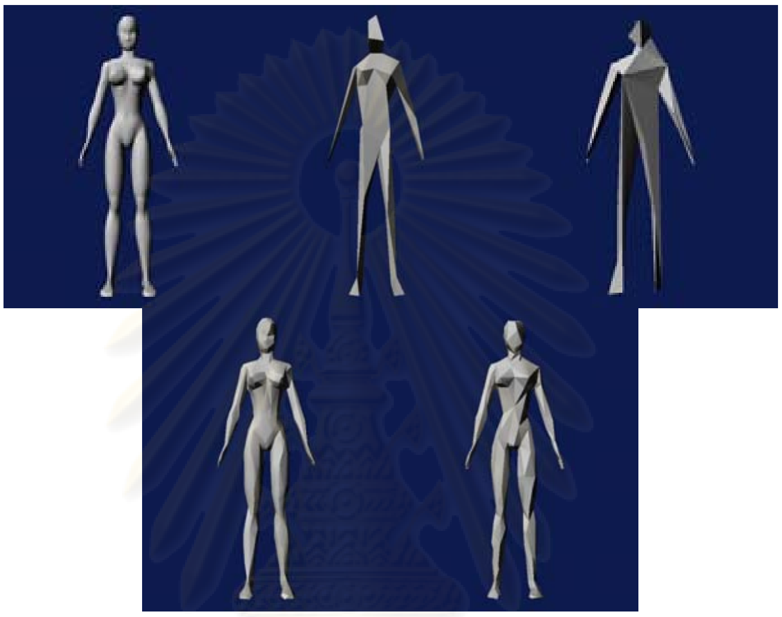
Table A-7: Running times after changing update parameters (cont'd)

Model	Parameters	50%	20%	10%	5%	2%	1%
351	*n*-8 layers	18.625	29.594	32.984	N/A[*]	N/A[*]	N/A[*]
	n-4 layers	28.953	51.531	56.359	N/A[*]	N/A[*]	N/A[*]
	$\|C\|\geq\|H\|\times 6$	21.234	36.172	39.984	N/A[*]	N/A[*]	N/A[*]
	$\|C\|\geq\|H\|\times 2$	20.969	38.188	41.797	N/A[*]	N/A[*]	N/A[*]
	No Cache	25.750	44.234	46.625	N/A[*]	N/A[*]	N/A[*]
	Regular	18.987	35.030	36.693	N/A[*]	N/A[*]	N/A[*]
361	*n*-8 layers	55.375	86.375	100.781	108.359	111.750	114.031
	n-4 layers	97.078	150.516	169.938	177.109	180.063	181.297
	$\|C\|\geq\|H\|\times 6$	45.609	84.797	99.250	104.406	108.203	109.797
	$\|C\|\geq\|H\|\times 2$	74.781	137.063	153.688	157.922	161.172	162.734
	No Cache	79.563	134.016	150.250	156.391	160.844	162.563
	Regular	49.631	88.770	93.965	97.891	100.955	102.267
371	*n*-8 layers	54.063	83.016	95.734	100.828	106.016	108.500
	n-4 layers	84.219	130.000	145.516	152.547	155.797	156.750
	$\|C\|\geq\|H\|\times 6$	40.906	76.938	90.125	94.531	98.141	99.297
	$\|C\|\geq\|H\|\times 2$	76.781	129.266	146.828	151.422	154.813	156.297
	No Cache	77.625	120.766	139.109	148.438	151.813	153.438
	Regular	58.394	87.265	98.652	104.530	107.044	108.426
381	*n*-8 layers	25.516	40.625	47.734	50.250	53.203	54.672
	n-4 layers	40.484	61.750	69.438	72.313	73.828	74.281
	$\|C\|\geq\|H\|\times 6$	21.297	38.563	45.313	47.609	49.094	49.969
	$\|C\|\geq\|H\|\times 2$	39.578	58.188	64.797	67.109	68.672	69.609
	No Cache	36.375	62.328	67.953	71.375	73.281	74.453
	Regular	28.150	46.206	50.483	52.846	54.398	55.430
391	*n*-8 layers	16.328	25.125	28.109	29.734	31.719	32.609
	n-4 layers	21.734	36.891	41.422	42.531	43.281	43.750
	$\|C\|\geq\|H\|\times 6$	12.547	22.984	25.875	27.766	28.516	29.453
	$\|C\|\geq\|H\|\times 2$	22.859	35.484	38.078	39.203	40.359	41.063
	No Cache	23.703	35.109	38.344	39.891	41.422	42.375
	Regular	14.551	22.643	25.266	26.448	27.810	28.571

[*] N/A: The algorithm exhausted all possible contractions before the given level

A.3 Pictures of results

For all figures: (Top row) Full model, 1% simplification with QEM, and our method; (Bottom row) 10% simplification with QEM, and our method.


(a)

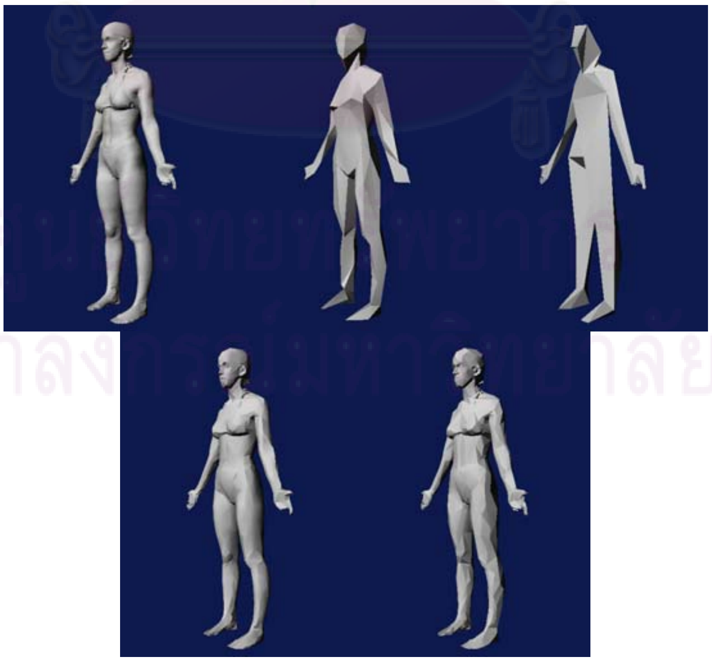
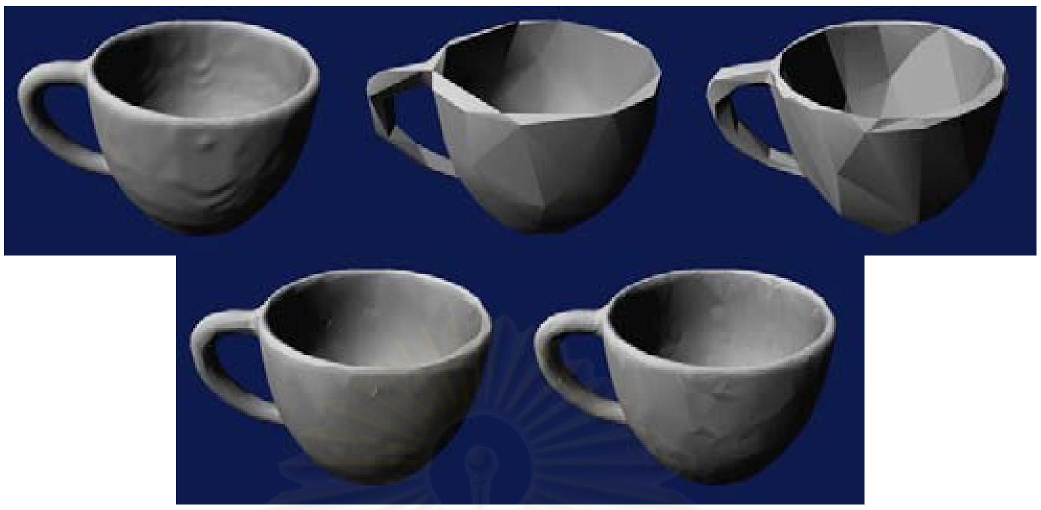
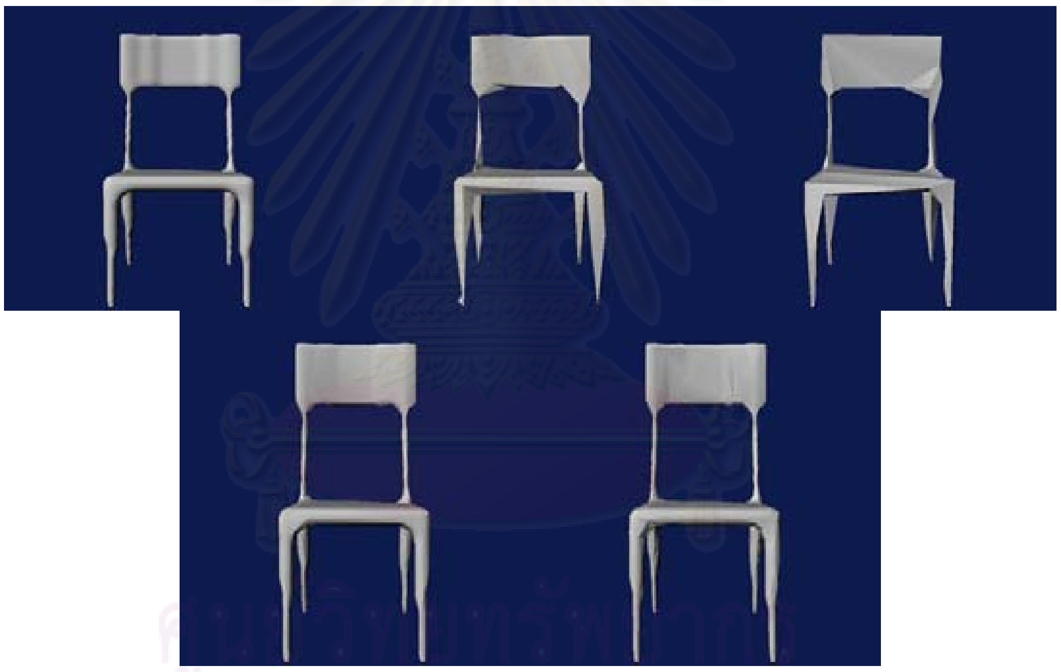

(b)

Figure A-14: 1% and 10% simplified models: (a) Female 1 and (b) Female 2


(a)


(b)

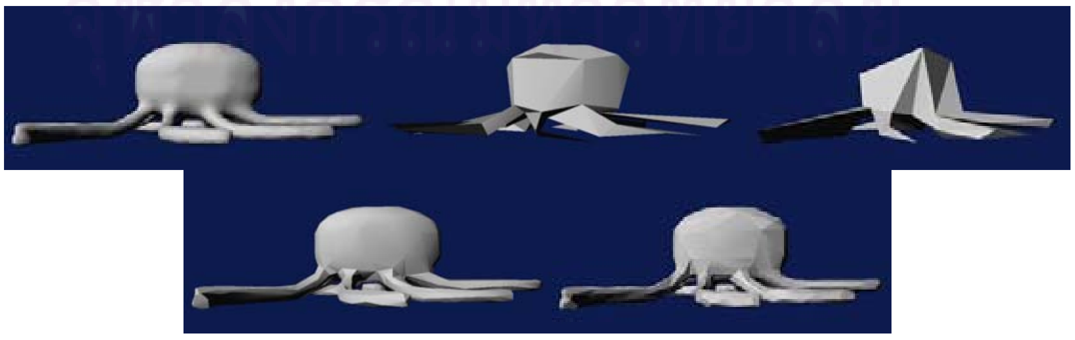
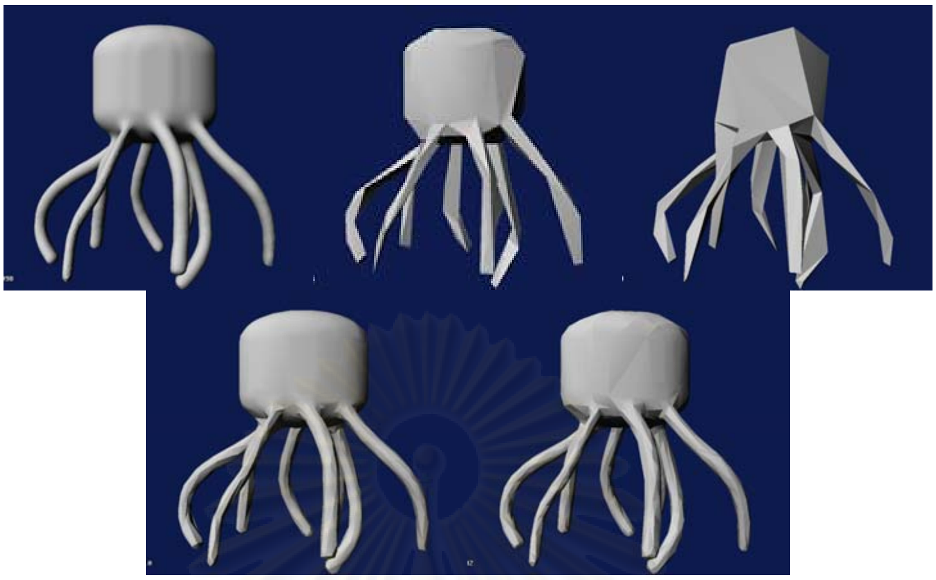
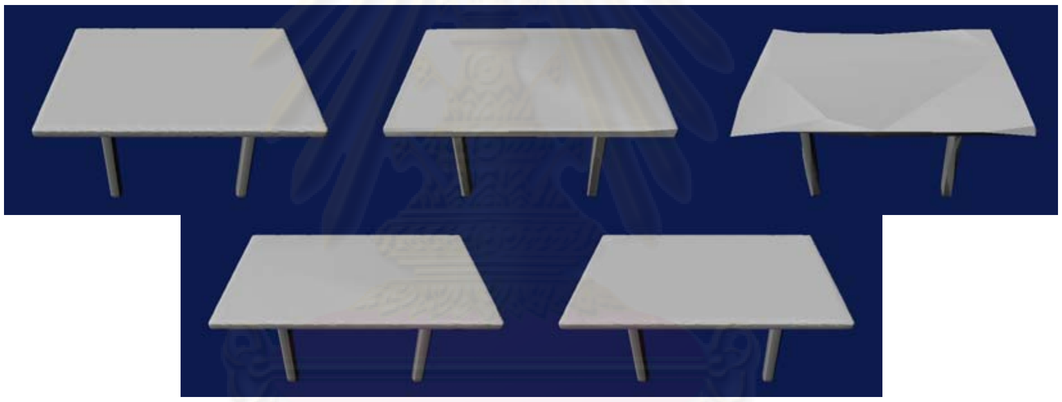

(c)

Figure A-15: 1% and 10% simplified models: (a) Cup, (b) Chair, (c) Squid


(a)


(b)

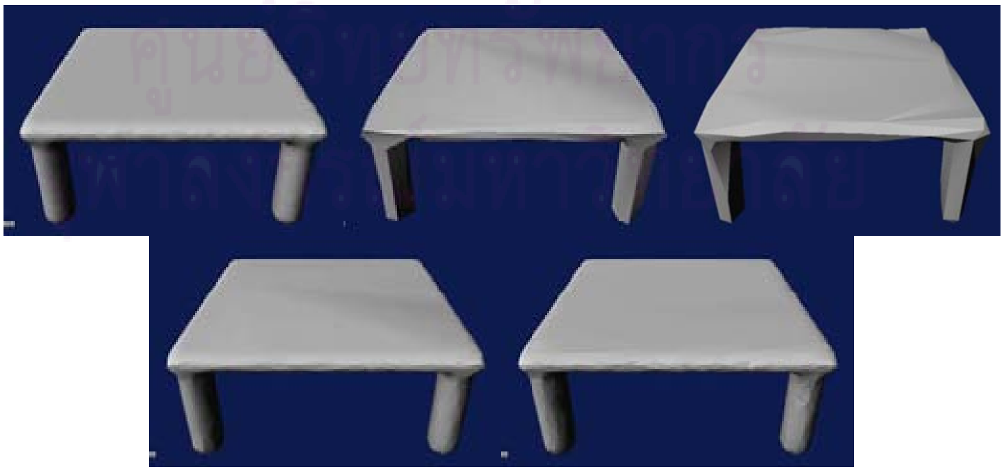
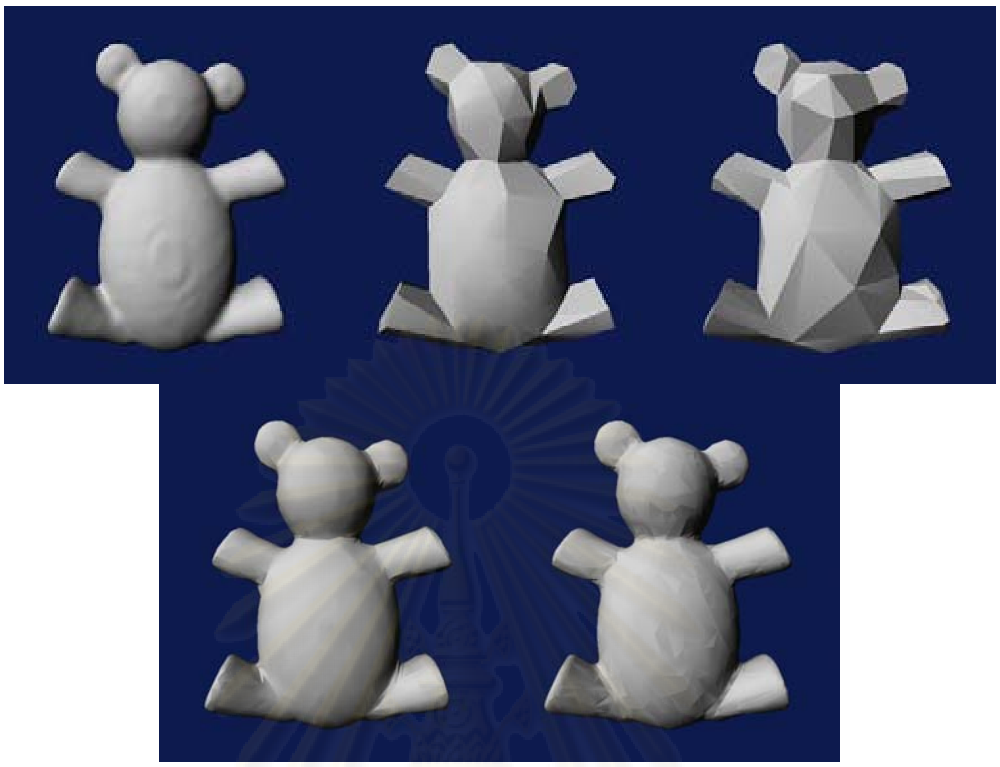
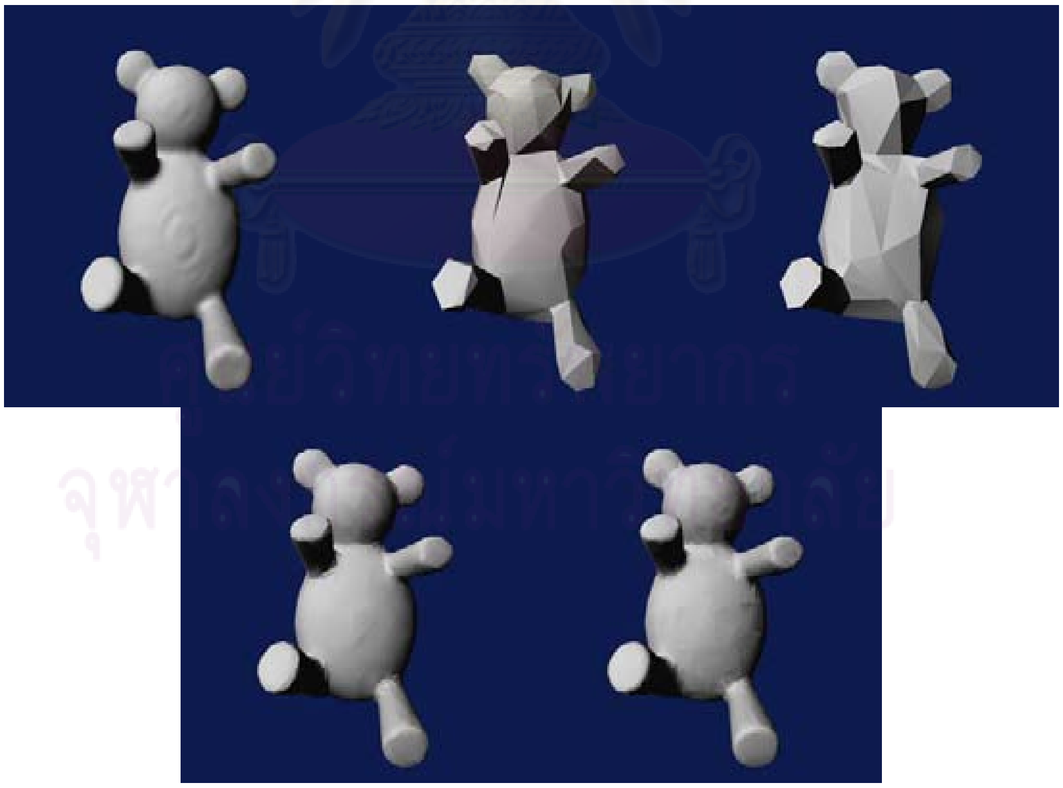

(c)

Figure A-16: 1% and 10% simplified models: (a) Squid 2, (b) Table 1, (c) Table 2

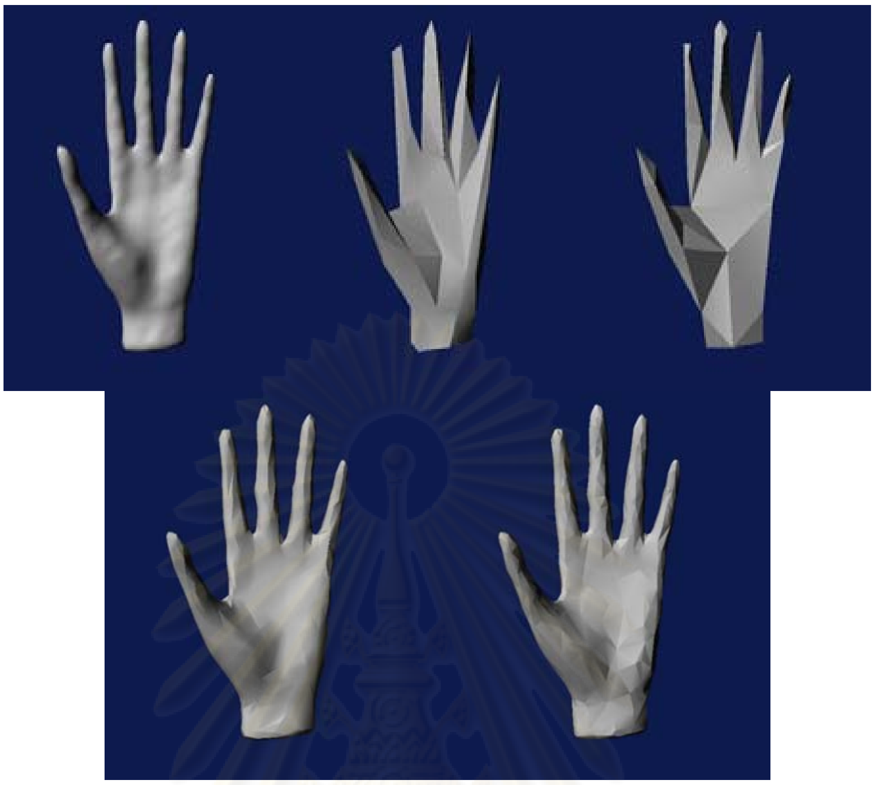


(a)



(b)

Figure A-17: 1% and 10% simplified models: (a) Teddy, (b) Teddy 2


(a)

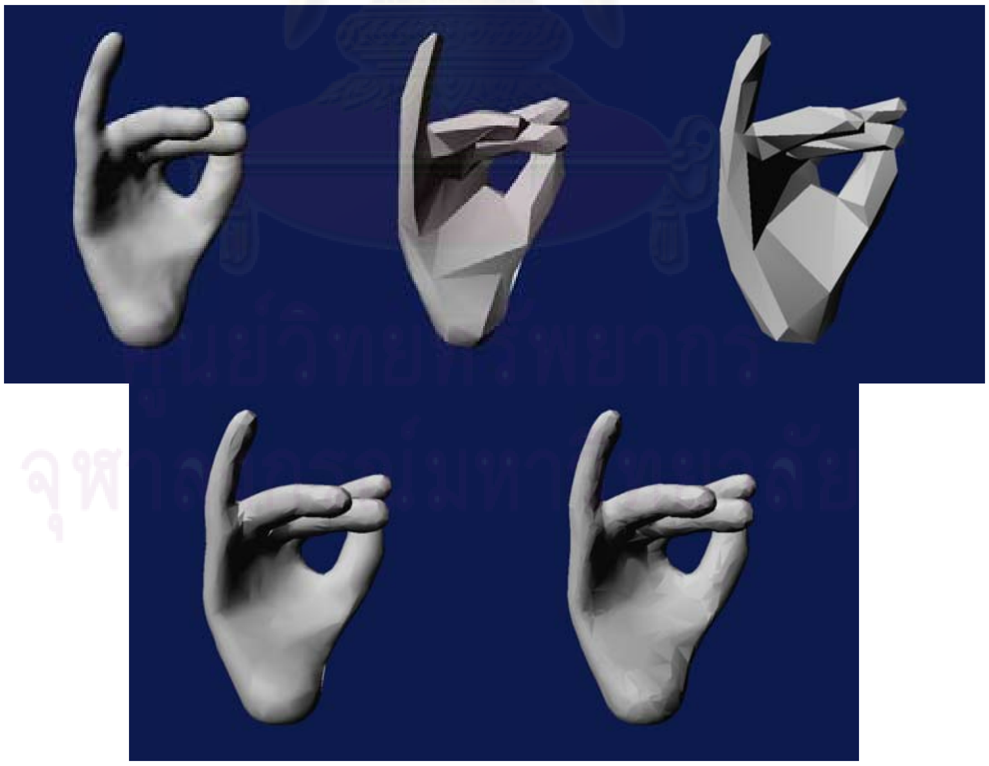
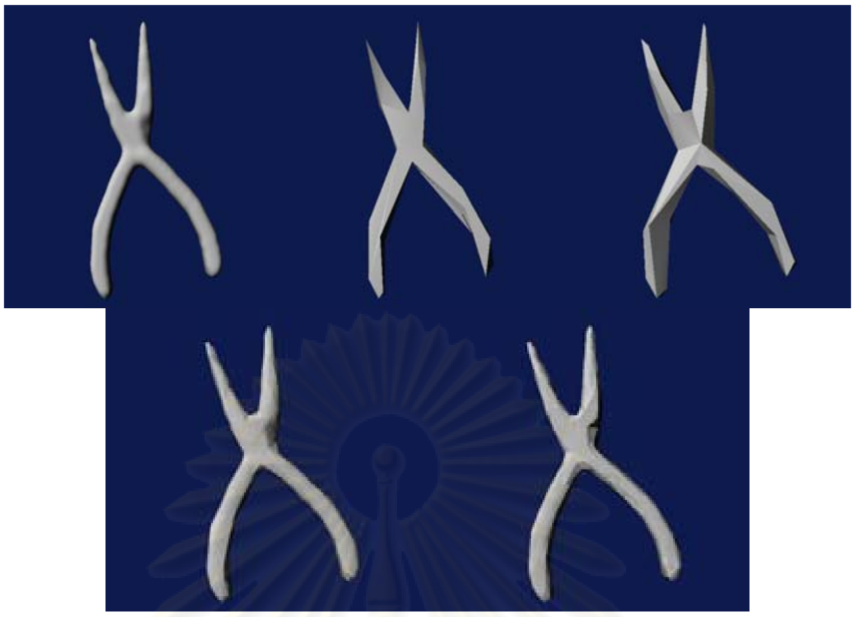
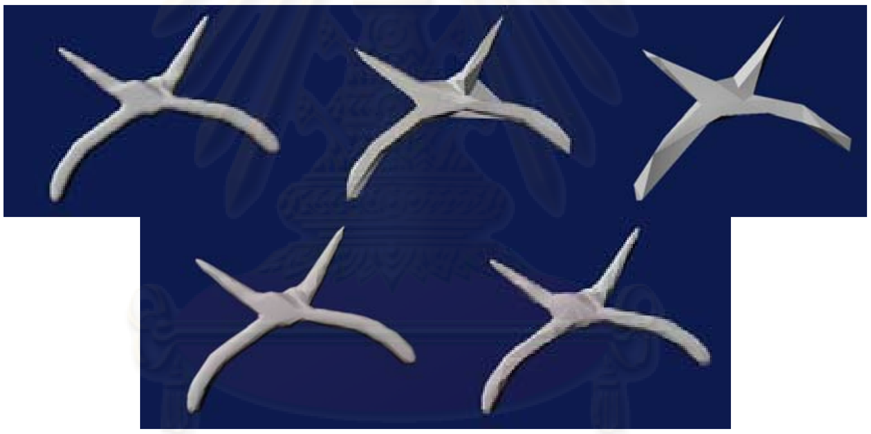

(b)

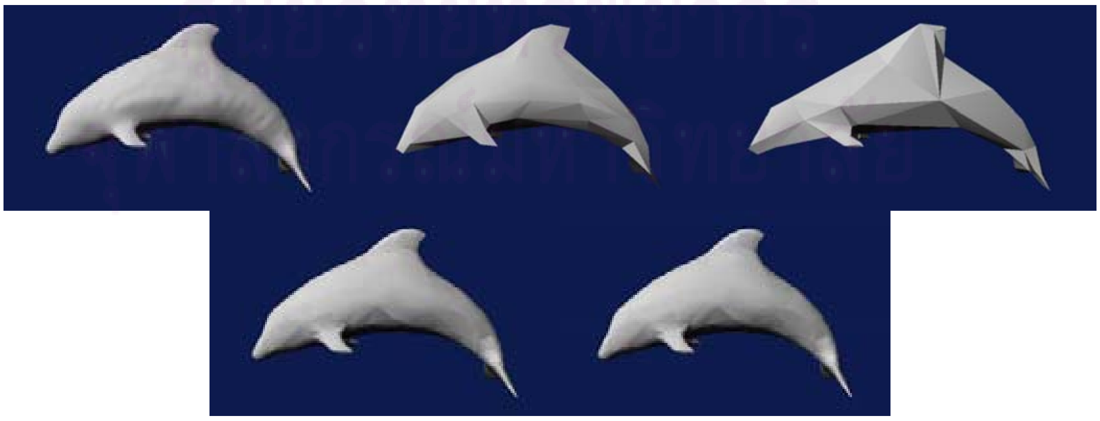
Figure A-18: 1% and 10% simplified models: (a) Hand, (b) Hand 2



(a)

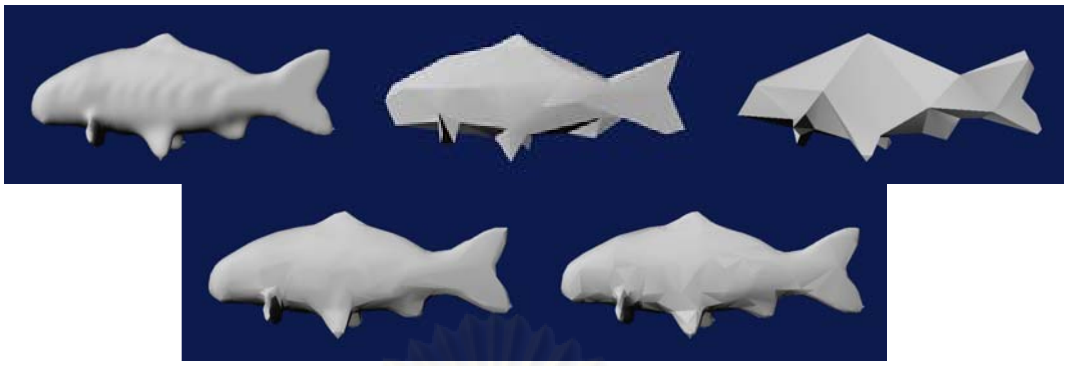


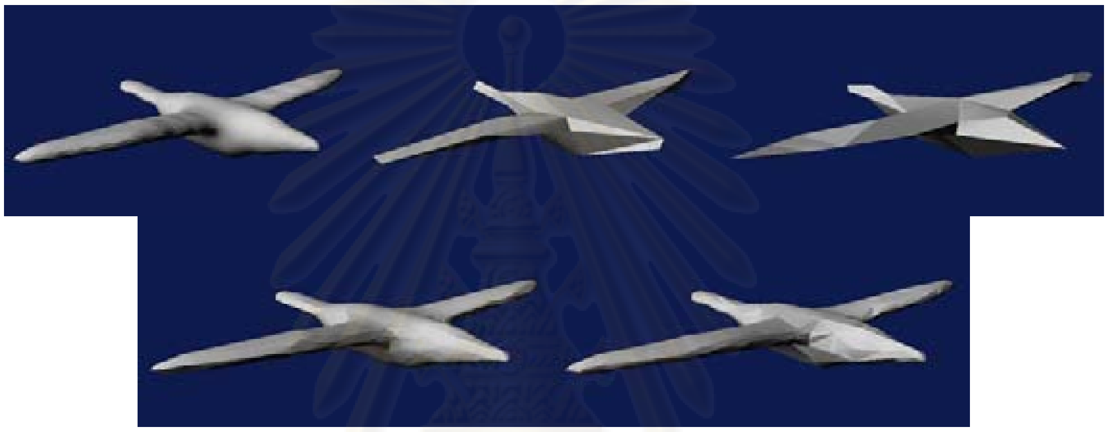
(b)



(c)

Figure A-19: 1% and 10% simplified models: (a) Pliers, (b) Pliers 2, (c) Dolphin


(a)


(b)

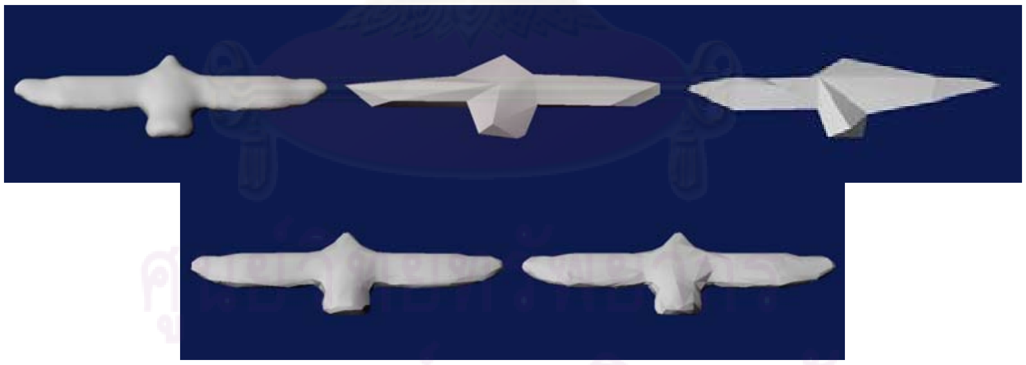
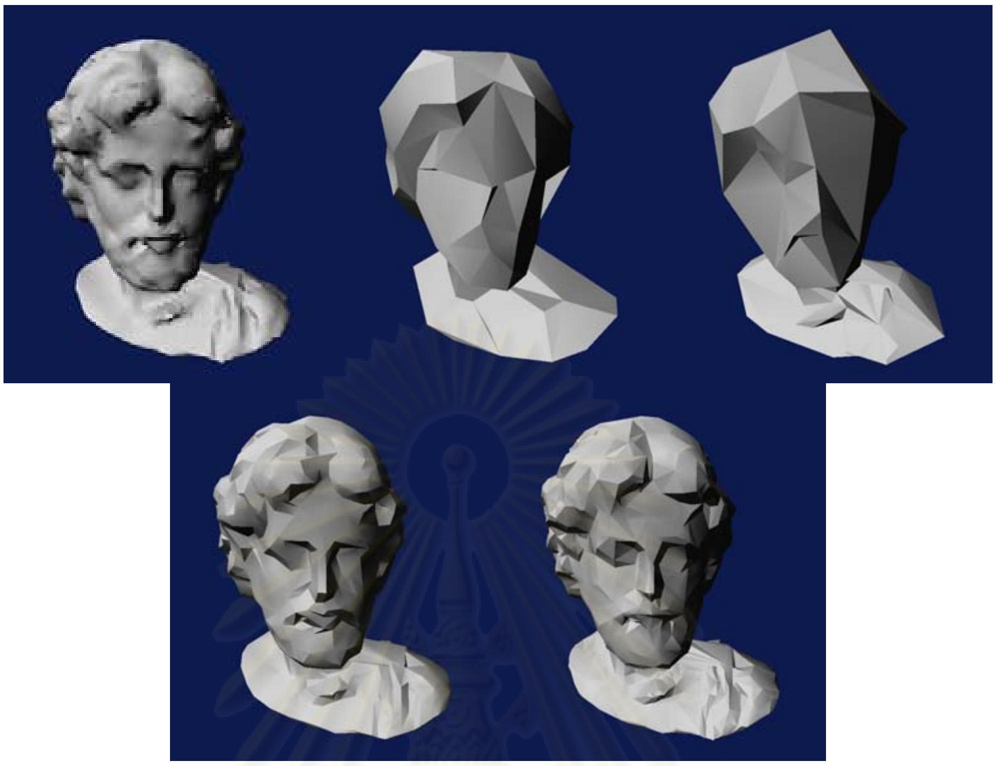
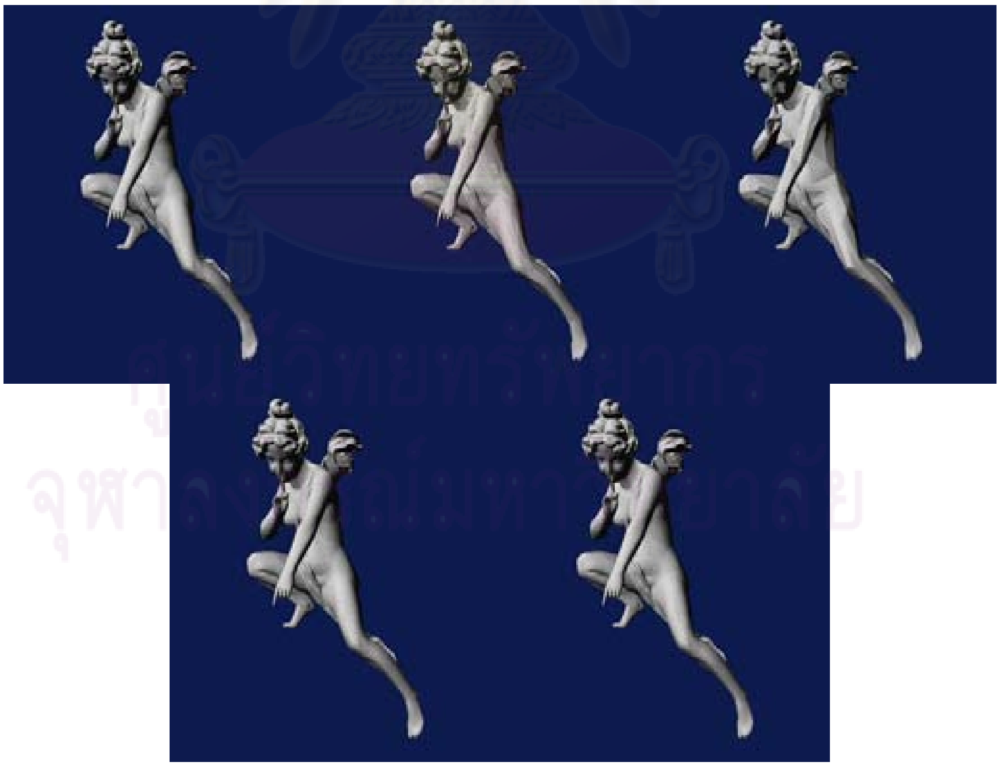

(c)

Figure A-20: 1% and 10% simplified models: (a) Fish, (b) Bird, (c) Bird 2

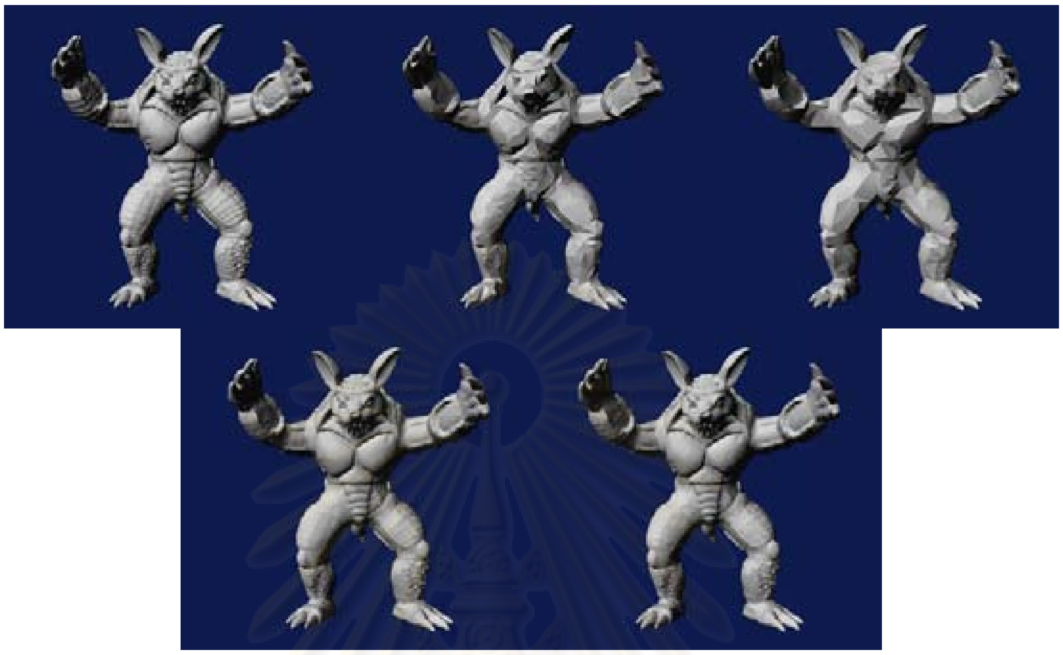


(a)

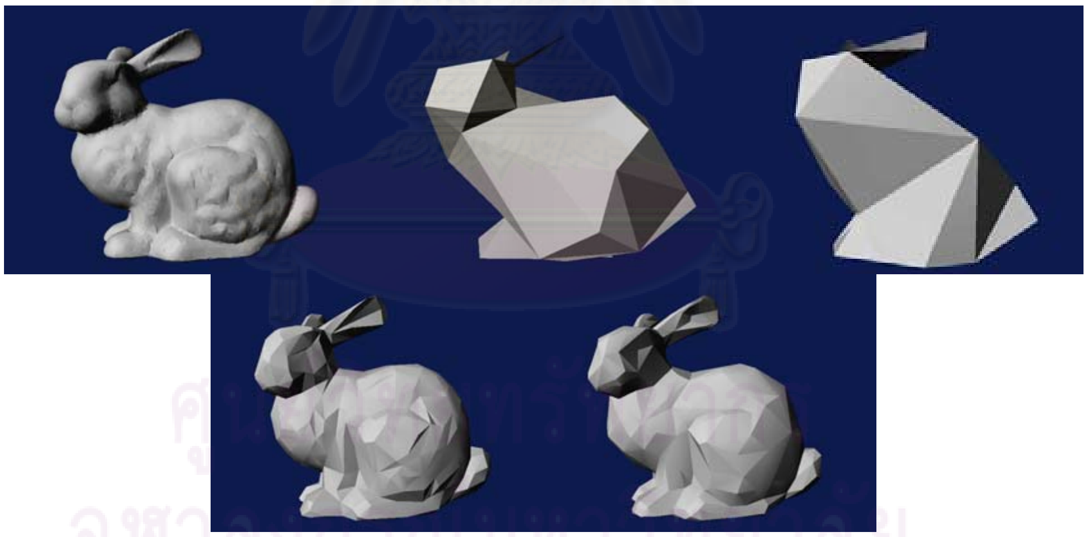


(b)

Figure A-21: 1% and 10% simplified models: (a) Head, (b) Angel

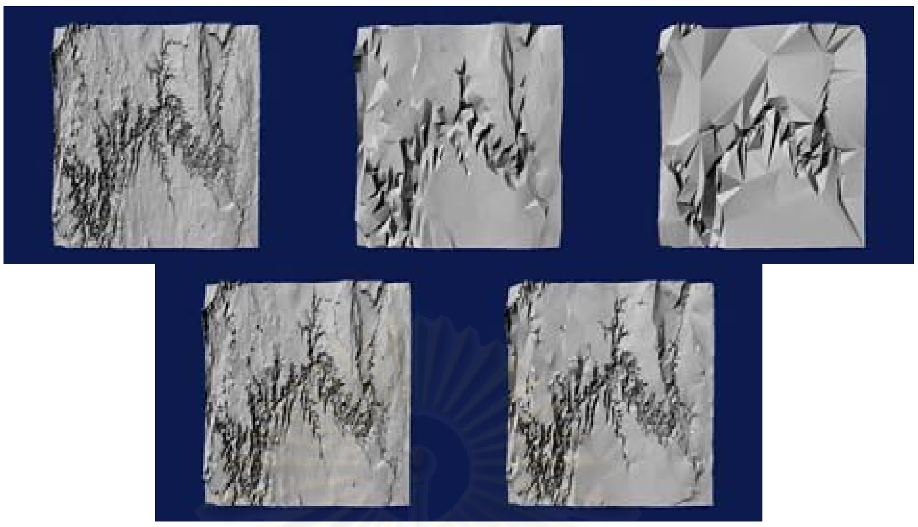


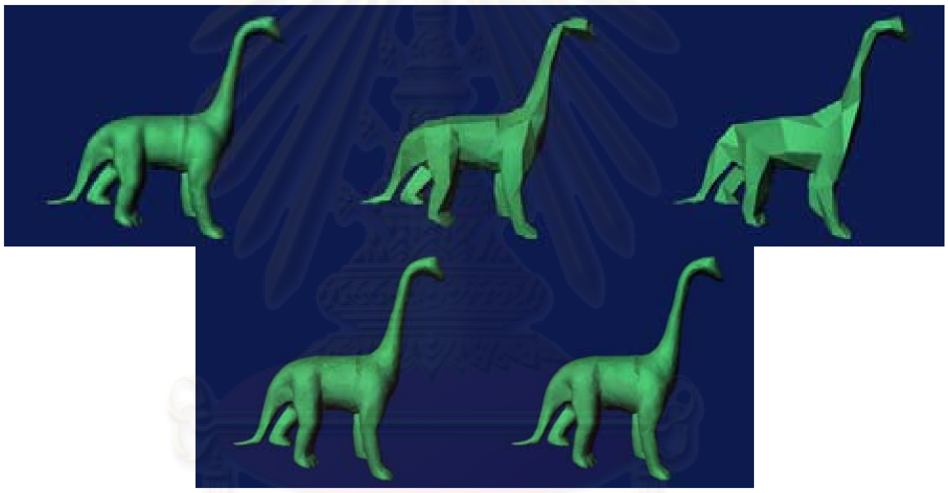
(a)



(b)

Figure A-22: 1% and 10% simplified models: (a) Big Armadillo, (b) Bunny


(a)


(b)

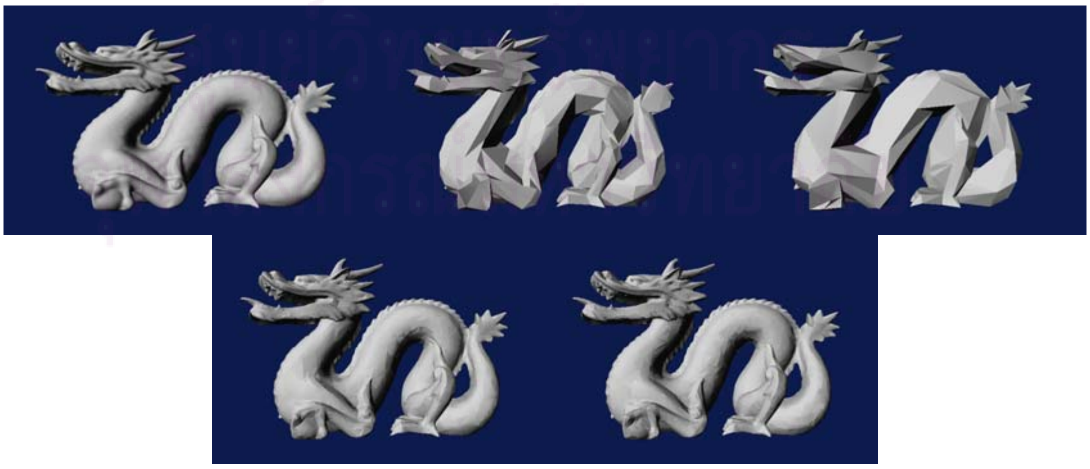
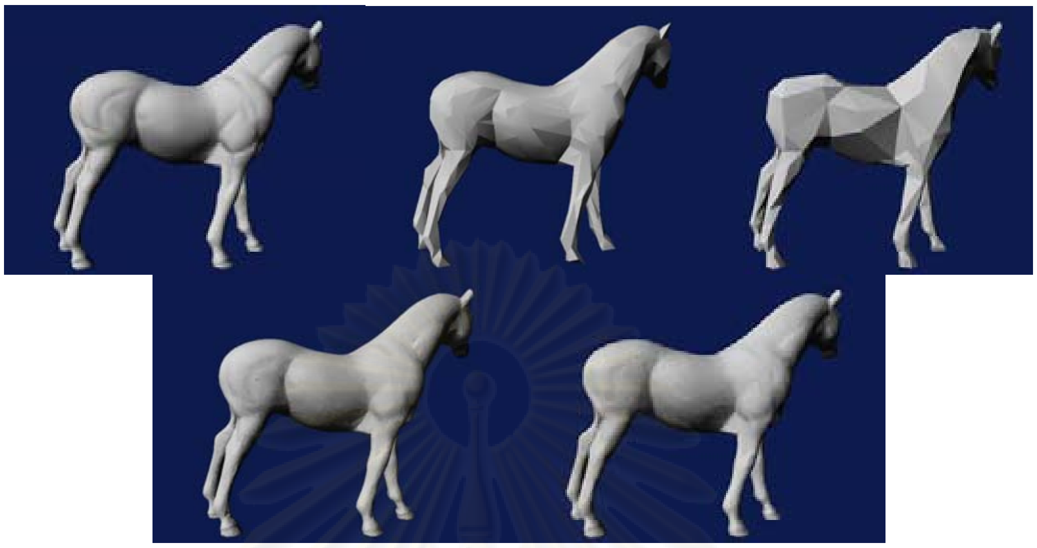

(c)

Figure A-23: 1% and 10% simplified models: (a) Canyon, (b) Dinosaur, (c) Dragon


(a)

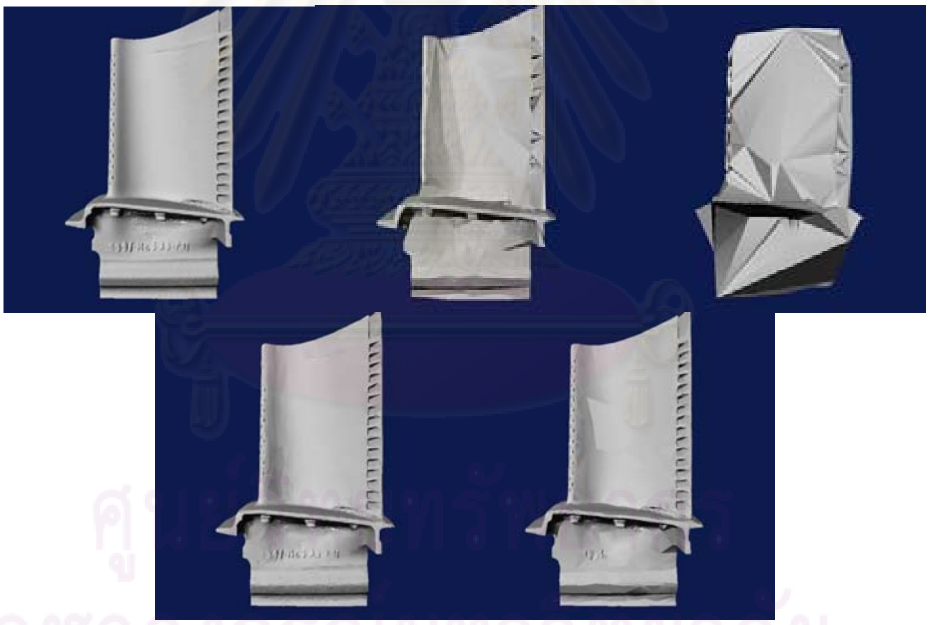

(b)

Figure A-24: 1% and 10% simplified models: (a) Horse, (b) Turbine

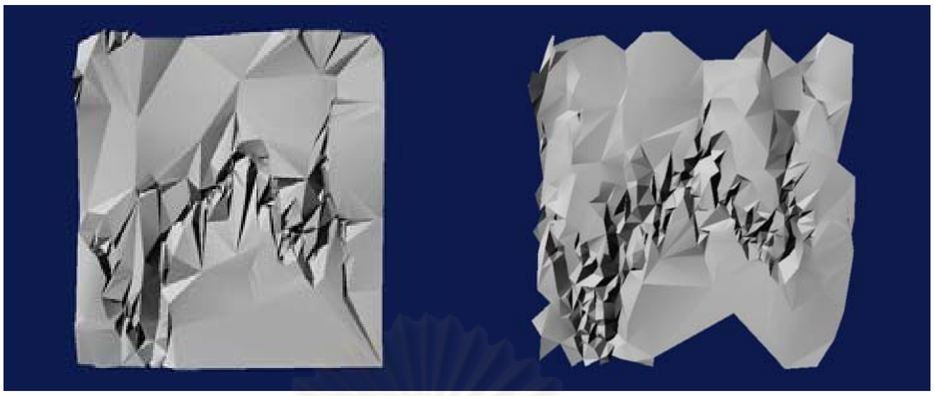


Figure A-25: Canyon model, simplified with (left) and without (right) boundary preservation

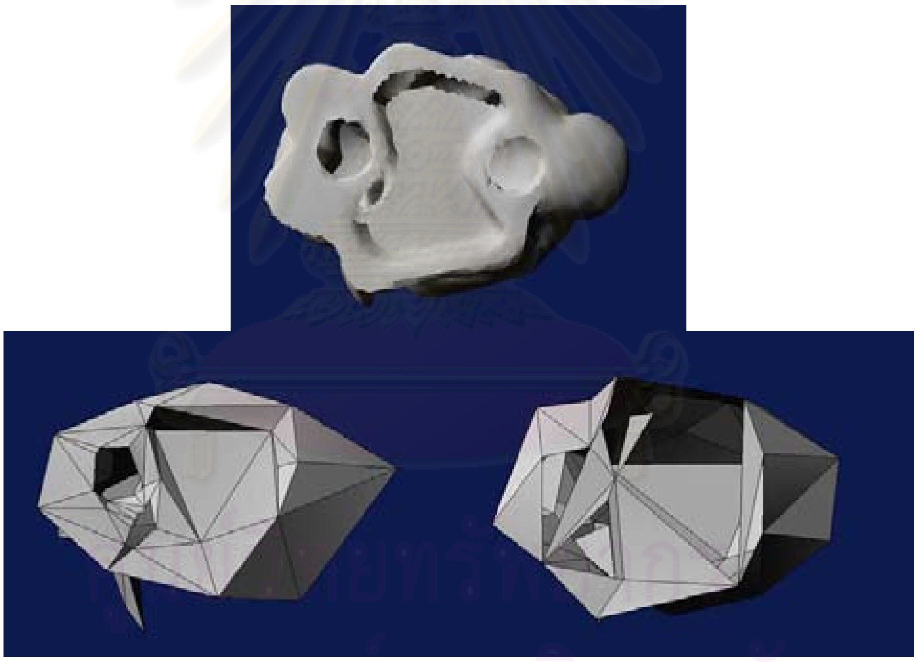


Figure A-26: Bottom of bunny model: Full (left), simplified with boundary preservation (middle), without (right)

BIOGRAPHY

Varakorn Ungvichian was born on 22 May, 1983. Ungvichian received a Bachelor of Engineering (B.Eng.) degree in Computer Engineering from the Faculty of Engineering, Chulalongkorn University in 2005, entered the Master of Engineering curriculum at the Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University the same year, received a Master of Engineering (M.Eng.) degree in 2007, and entered the Doctor of Engineering curriculum at the Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University in 2007.

Ungvichian has presented two of his research papers at international level conferences: "Mapping a 3-D Model into Abstract Cellular Complex Format" at CAD '06 in Phuket, Thailand from June 19-23, 2006, and "Quadrangle Collapse Mesh Reduction with Regularity and Angular Deviation Bias" at ICCMS 2010 in Sanya, China from January 22-24, 2010.