Accurate 3D Mesh Simplification

Elena Ovreiu

Numéro d'ordre:2012-ISAL-0145 Année 2012

THÈSE

Présentée devant

L'INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE LYON

pour obtenir

LE GRADE DE DOCTEUR

École doctorale: Électronique, Électrotechnique, Automatique
Formation doctorale: Images et Systèmes

En cotutelle avec

UNIVERSITATEA POLITEHNICA BUCURESTI
Spécialité : Ingénierie Electronique et Télécommunications

par

Elena OVREIU

Accurate 3D Mesh Simplification

Soutenue le 12 décembre 2012 devant la commission d'examen

Jury:

Marc Antonini	Directeur de recherche CNRS, I3S, Sophia Antipolis	Rapporteur
Titus Zaharia	Professeur, ARTEMIS, INT, Paris	Rapporteur
Alfred Bruckstein	Professeur, GIP, Technion, Haifa, Israel	Examinateur
Florent Dupont	Professeur, LIRIS, Univertisté Claude Bernard de Lyon 1	Examinateur
Mihai Ciuc	Maître de Conférence, LAPI, Politehnica, Bucarest, Roumanie	Examinateur
Sébastien Valette	Chargé de recherche CNRS, CREATIS Lyon	Co-encadrant
Radu Dogaru	Professeur, NHPCL, Politehnica, Bucarest, Roumanie	Co-directeur de thèse
Rémy Prost	Professeur, CREATIS, INSA, Lyon	Co-directeur de thèse

Acknowledgements

The completion of my dissertation was a long journey with good or less good moments and I would not have been able to complete this journey without the aid and support of some people to whom I would like to express my gratitude.

First of all, I express my gratitude to my advisors, Professor Remy Prost and Sebastien Valette for their support, guidance and advices throughout my PhD research.
Furthermore, I am very grateful to Professor Radu Dogaru, who accepted to be the advicer of my PhD thesis from Politehnica University of Bucharest.
My thanks also go to the members of my PhD committe for their suggestions and advices regarding my thesis.

Thanks to all my colleagues in Creatis, where I have spent the last 3 years. Xavier Lojacono and Remy Blanchard, *merci beaucoup* for your help with French. Thank you, Xavier for helping and encouraging me in the difficult moments during my stay in France. Pierre Ferrier, thank you for the countless times you have repaired my computer.
I would also like to thank to Eduardo Davila for his help and precious advices regarding my PhD and my future career, as well.
Many thanks to Professor Maciej Orksiz who gave me the opportunity for my scientific assignment in Bogota and to Isabelle Magnin, the head of Creatis, who always supported me.

I would like to thank to all my friends in France for the nice moments we spent together. Asma, you were my family in France. Thank you for cooking for me tajin and couscous, for your delicious cookies and the turkish coffee, for your smile and your daily good mood which was transmitted to me, too. Ioana and Mihnea, thank you for preparing polenta for each *soirée* organized in my flat, thank you for introducing me in the world of climbing.
Many thanks to Juan Gabriel Riveros for his patience regarding my code, for many things he taught me, for his daily countless "Girl, I hate you!" and for the delicious salad he prepared. Thank you for the accomodation in Bogota. Thanks for everythink! You will forever be one of the special persons in my life.

My sincere thank goes to the group LAPI, especially to Alina Sultana and Mihai Ciuc. Thank you for your advices, your support and help during my Phd and afterwards. LAPI has always been my second home.

I would also like to thank to many special people who helped me during my Phd. Many thanks to Professor Craig Gotsman for giving me the oportunitty to work with him and to spend one month within his research group.
My thank also goes to Professor Jean-Marie Becker and Damien Rohmer from CPE for many hours spent to help me with my thesis research, for their insightful comments and

advices.

My sincere gratitude goes to Professor Freddy Bruckstein and to his wife, Rita for the precious things I have learned from you. You always have made me feel as belonging to your family and you made my research visit to Israel unforgettable.

Andreea, thanks for being next to me during these three years through all the good and bad moments. We have laughed and cried, traveled and shopped together.

Thank you, Costin for everything you have done for me in the beginning of my PhD. Thanks for your support and advices, for the patience to listen to all my problems and get involved in solving them, for the patience to follow step by step the evolution of my PhD.

Thank you, Dan for your help in writing my PhD thesis. You have spent many hours to understand and rewrite this thesis.

For all the fun I have had in the half part of my Phd, Noemi was responsible. Thank you, sis.

Cosmin, you offered me the moral support and the equilibrium I needed to complete my Phd. And I am deeply grateful to you.

No acknowledgments would be completed without giving thanks to my family: my sister, my parents and my grandparents who always supported me in all my pursuits. Without you and your love I would not be the person I am today.

Last, but certainly not least, I will forever be thankful to my advisor, model and mentor, Professor Vasile Buzuloiu. His dedication to and love for his students, his enthusiasm for science determined me to pursue a career in research.
Even if he is not longer with us, I hope he is proud for my PhD defence.

Accurate 3D Mesh Simplification

Complex 3D digital objects are used in many domains such as animation films, scientific visualization, medical imaging and computer vision. These objects are usually represented by triangular meshes with many triangles. The simplification of those objects in order to keep them as close as possible to the original has received a lot of attention in the recent years.

In this context, we propose a simplification algorithm which is focused on the accuracy of the simplifications. The mesh simplification uses edges collapses with vertex relocation by minimizing an error metric. Accuracy is obtained with the two error metrics we use: the Accurate Measure of Quadratic Error (AMQE) and the Symmetric Measure of Quadratic Error (SMQE).

AMQE is computed as the weighted sum of squared distances between the simplified mesh and the original one. Accuracy of the measure of the geometric deviation introduced in the mesh by an edge collapse is given by the distances between surfaces. The distances are computed in between sample points of the simplified mesh and the faces of the original one.

SMQE is similar to the AMQE method but computed in the both, direct and reverse directions, i.e. simplified to original and original to simplified meshes. The SMQE approach is computationnaly more expensive than the AMQE but the advantage of computing the AMQE in a reverse fashion results in the preservation of boundaries, sharp features and isolated regions of the mesh.

For both measures we obtain better results than methods proposed in the literature.

Simplification précise de maillages 3D

Les objets numériques 3D sont utilisés dans de nombreux domaines, les films d'animations, la visualisation scientifique, l'imagerie médicale, la vision par ordinateur.... Ces objets sont généralement représentés par des maillages à faces triangulaires avec un nombre énorme de triangles. La simplification de ces objets, avec préservation de la géométrie originale, a fait l'objet de nombreux travaux durant ces dernières années.

Dans cette thèse, nous proposons un algorithme de simplification qui permet l'obtention d'objets simplifiés de grande précision. Nous utilisons des fusions de couples de sommets avec une relocalisation du sommet résultant qui minimise une métrique d'erreur. Nous utilisons deux types de mesures quadratiques de l'erreur : l'une uniquement entre l'objet simplifié et l'objet original (Accurate Measure of Quadratic Error (AMQE)) et l'autre prend aussi en compte l'erreur entre l'objet original et l'objet simplifié ((Symmetric Measure of Quadratic Error (SMQE)) . Le coût calculatoire est plus important pour la seconde mesure mais elle permet une préservation des arêtes vives et des régions isolées de l'objet original par l'algorithme de simplification. Les deux mesures conduisent à des objets simplifiés plus fidèles aux originaux que les méthodes actuelles de la littérature.

Simplification précise de maillages 3D

Les objets numériques 3D sont utilisés dans de nombreux domaines, les films d'animations, la visualisation scientifique, l'imagerie médicale, la vision par ordinateur.... Ces objets sont généralement représentés par des maillages à faces triangulaires avec un nombre énorme de triangles. La simplification de ces objets, avec préservation de la géométrie originale, a fait l'objet de nombreux travaux durant ces dernières années.

Dans cette thèse, nous proposons un algorithme de simplification qui permet l'obtention d'objets simplifiés de grande précision. Nous utilisons des fusions de couples de sommets avec une relocalisation du sommet résultant qui minimise une métrique d'erreur. Nous utilisons deux types de mesures quadratiques de l'erreur : l'une uniquement entre l'objet simplifié et l'objet original (Accurate Measure of Quadratic Error (AMQE)) et l'autre prend aussi en compte l'erreur entre l'objet original et l'objet simplifié ((Symmetric Measure of Quadratic Error (SMQE)) . Le coût calculatoire est plus important pour la seconde mesure mais elle permet une préservation des arêtes vives et des régions isolées de l'objet original par l'algorithme de simplification.

AMQE représente la somme pondérée des carrés des distances entre l'objet simplifié et l'objet original. Nous utilisons comme poids de l'aire de triangles. De cette façon, les distances de triangles plus grandes sont plus importantes dans l'erreur finale.

Afin d'avoir une mesure plus précise des distances, chaque triangle de l'objet simplifié est subdivisé, itérativement, en quatre triangles (1:4). Le nombre de subdivisions 1:4 est déterminé de manière à maintenir la proportionnalité entre le nombre de triangles de l'objet simplifié et d'objet original. Par cette subdivision, un ensemble de points d'échantillonnage est généré pour chaque triangle de l'objet simplifié. La distance entre un triangle de l'objet simplifié et de l'objet original est calculée en prenant en considération les distances aux points d'échantillonnage.

PQP mesure la distance d'un point à un triangle de l'objet original et non aux plans associés aux triangles (comme QEM) nous obtenons ainsi une évaluation précise de l'écart géométrique entre deux objets.

Pour préserver des régions isolées de l'objet original, nous proposons d'utiliser SMQE qui prend en compte l'erreur entre l'objet original et l'objet simplifié.

SMQE est calculée avec AMQE, mais du fait que le coût calculatoire est plus important pour cette mesure, nous proposons certaines approximations.

Après chaque étape de simplification, au lieu de réévaluer les distances de chacun des triangles de l'objet original, nous déterminons les régions de l'objet original et de l'objet simplifié qui sont les plus touchées par la simplification. De cette façon, nous limitons la ré évaluation des distances uniquement pour ces régions.

Les deux mesures conduisent à des objets simplifés plus fidèles aux originaux que les méthodes actuelles de la littérature.

Notre méthode utilise des fusions de couples de sommets pour obtenir les simplifications.

Le sommet résultant est placé en un point de l'espace qui minimise la métrique connue dans la littérature sous le nom : quadratic error metric (QEM).

QEM utilise les matrices de formes quadratiques, associées à chaque triangle de l'objet, pour calculer l'erreur. Les sommets sont la solution d'un système d'équations linéaires. Quand le sommet est déplacé, le coût de déplacement est considéré comme la somme des carrés des distances du sommet aux plans tangents aux triangles adjacents.

Sur la base de cette méthode, le sommet résultat par la fusion d'arêtes est placé de manière à minimiser cette somme. Nous utilisons cette méthode pour localiser le sommet résultant car elle est très rapide.

La méthode de simplification utilise de manière itérative les fusions des aêtes jusqu'à ce que la condition d'arrêt soit atteinte. Dans notre algorithme, la condition d'arrêt est le nombre de sommets de l'objet simplifié fixé par l'utilisateur.

Pour les résultats expérimentaux nous avons utilisé les modèles suivants : Pieta, Bunny, Octa-flower, Beethoven, Dragon, Bones, Venus, Horse.

Pour évaluer les erreurs introduites dans la simplification par nos méthodes, nous utilisons la distance de Hausdorff et de l'erreur quadratique. Nous mesurons la distance de Hausdorff à l'aide du logiciel Metro.

Nous avons aussi comparé les erreurs, mesurées par la distance de Hausdorff et la distance quadratique, pour nos simplifications, avec les celles obtenues avec QEM. Pour tous les modèles, nous avons obtenu de meilleurs résultats que QEM avec la distance de Hausdorff et la distance quadratique.

La mesure d'erreur utilisée par QEM, pour évaluer l'écart géométrique introduit par une fusion de couples de sommets, est la distance entre le sommet et les plans des triangles associés au sommets. La distance calculée au plan support du triangle est différente de la distance calculée sur les triangles. Cette différence est plus importante pour les surfaces courbes.
Notre méthode utilise la distance entre le sommet et le triangle, elle est donc plus précise.

L'inconvénient de notre méthode est sont coût en temps d'exécution. Elle peut prendre plus dún jour pour simplifier un objet avec 50 000 sommets. La complexité est générée par PQP.

Parce ce que PQP n'est pas un structure dynamique, chaque fois que nous modifions une petite région de l'objet, la structure doit être reconstruite. La complexité n'est pas critique parce que nos simplifications sont obtenues hors ligne, notre objectif est d'obtenir des simplifications caractérisé par un haut niveau de précision.

Contents

Abstract **v**

Contents **x**

1 General Introduction **1**
1.1 Introduction . . . 1
1.2 Motivation . . . 1
1.3 Contributions . . . 4
1.4 Overview of the dissertation . . . 4

2 Surface Representation **7**
2.1 3D Digital Objects . . . 7
2.2 Surface Representation vs. Volumetric Representation . . . 8
2.3 Surface Representation . . . 10
2.3.1 Parametric Representation . . . 11
2.3.2 ImplicitRepresentation . . . 19

3 State of the Art of Triangular Mesh Simplification **21**
3.0.3 Hausdorff Distance . . . 21
3.0.4 Quadratic Error . . . 22
3.1 Metrics in practice for Mesh Simplification . . . 22
3.1.1 Vertex-to-Vertex Distance . . . 23
3.1.2 Vertex-to-Plane Distance . . . 23
3.1.3 Surface-to-Surface Distance . . . 26
3.2 Mesh Simplification Operations . . . 35
3.2.1 Iterative Simplification . . . 36
3.2.2 Direct Simplification . . . 42

4 Contributions **49**
4.1 Proposed Simplification Algorithm: overview . . . 52
4.2 Edge Collapse Simulations . . . 55
4.3 First Error Metric used: Accurately Measured Quadratic Error . . . 56
4.4 Second Error Metric used: Symmetric Measure of Quadratic Error . . . 66
4.5 QEM-based Vertex Optimization . . . 70
4.6 Volume-based Vertex Optimization . . . 75

5 Results **81**
5.1 Sharp Features Preservation . . . 81
5.2 Boundary Preservation . . . 100

Conclusions and perspectives **117**

Appendix **121**

A Appendix 1 **121**

B Appendix 2 **127**

Bibliography **139**

Personal Bibliography **139**

Chapter 1

General Introduction

1.1 Introduction

3D digital objects have become more present in our lives in the last two decades. They are widely used in many domains including medical visualisation, architectural and industrial design, virtual reality, cartography, remote sensing. With the increasing interest in 3D digital objects, the techniques for producing these objects have been improved and produce nowadays digital objects with millions, even billions of elements.
Complex objects simulate the reality very well but have as disadvantages difficulty in handling, rendering or transmitting over the internet. Moreover, the storage memory is larger for complex objects. For these reasons, mesh simplification is desirable.
The goal of the mesh simplification algorithms is to reduce the complexity of a mesh while preserving a high fidelity of the original during simplification.
In this context, this thesis is focused on reducing the complexity of digital objects. The goal is to create approximations of the original object with fewer elements but maintaining a high fidelity of the original (Figure 1.1).

1.2 Motivation

During the last few years developments in 3D acquisition techniques have permitted the generation of digital objects with a huge number of elements. On the one hand, the complexity means objects more realistic with a multitude of details. On the other hand, the complexity means a lot of information which makes the interaction with the object more difficult. In addition, these objects require more memory for storage.

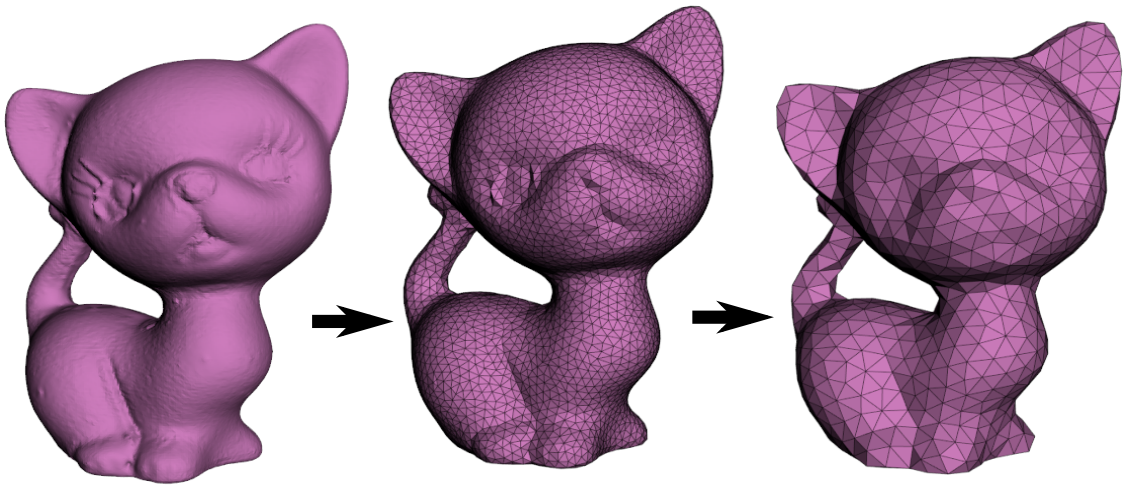


Figure 1.1: *A 3D object with two approximations. From left to right: The original model with 274 196 triangles, approximation with 10 000 triangles and 2000, respectively. Kitten model is provided courtesy of Frank ter Haar by the AIM@SHAPE Shape Repository*

In certain applications complexity is not necessary and a simplified versions of these objects can be used successfuly.
For example, in animated films it is important to have more details for the objects closer to the view point (see the characters in the *Shrek* animation film from Figure 1.2), while details are completely useless for objects far away from the view point, such as trees in the background.

In interactive applications such as video games or aircraft simulators, interactivity is more important than the visual quality of the objects. Thus, in these applications, it is necessary to reduce the complexity of the objects, as less information permits faster rendering and interactivity.
There are some 3D acquisition techniques which create objects with redundant information. As an example, the digital statue of Iulius Caesar (Figure 1.3) was created by a 3D laser scanner and contains approximatively 0.4 M vertices. We can easly tell that there are some flat regions such as the forehead or nose which can be approximated with fewer triangles without affecting the quality of the object.
In medical visualisation, the 3D digital reconstruction of the human organs is realised by extracting isosurfaces with *marching cubes* algorithms ([Lorensen and Cline, 1987]) from MRI (Magnetic Resonance Imaging) or CT (Computed Tomography) datasets. *Marching cubes* algorithms produce meshes with redundant information (Figure 1.3). To improve work with this kind of objects, simplification is required.
In cartography, elements such as rivers or roads can be represented with fewer details.

In all applications presented above, the complexity of the objects can be reduced in such a manner as to preserve the details and characteristics of the original. Among the ad-

Figure 1.2: *Synthetic objects used in production of animation films. The image, taken from the animation film* Shrek, *contains more details for the characters, because they are in the foreground and fewer details for the background.*

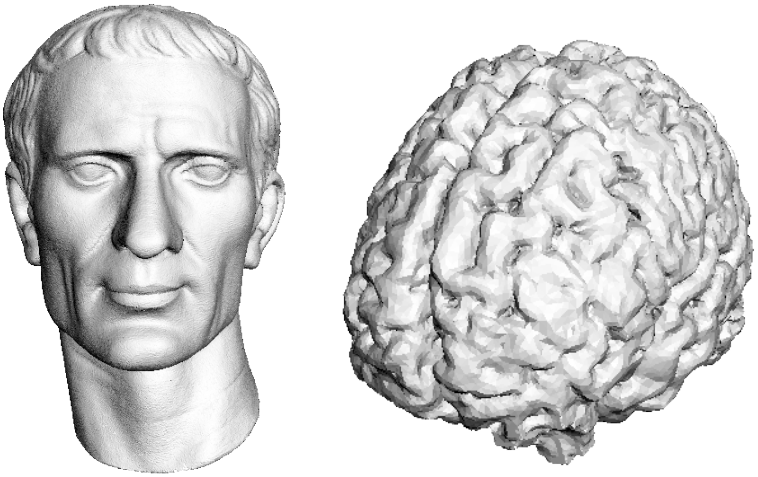


Figure 1.3: Left: The digital statue of Iulius Caesar, created by a 3D laser scanner. Right: A brain model created using marching cubes algorithm. The both models are provided courtesy of INRIA by the AIM@SHAPE Shape Repository.

vantages of reducing complexity are: a decrease in rendering time, increased rapidity of object manipulation and less memory used for objects storage. Also, transmission over internet is improved in terms of rapidity. Simplified objects are better adapted for different bandwidths and can be easier rendered on any destination workstation.

1.3 Contributions

This thesis proposes two different error metrics to measure the geometric deviation introduced by an edge collapse during simplification.

First, we propose a simplification algorithm which uses an accurately measured one-sided quadratic error metric (AMQE).This error metric provides an accurate measure of the geometric deviation between the mesh after an edge collapse and the original object. The faces of the mesh are sampled and an exact distance is computed from the sample points to the original mesh. The accurate quadratic error will be the sum of the weighted squared distances. Because the distances from the sample points to the original mesh are accurately computed, this error metric provides an accurate characterization of the geometric deviation introduced by an edge collapse. More than that, the error metric provides a global measurements of the geometric deviation between the simplified and original meshes.
The accurate one-sided quadratic error metric produces simplifications with a high preservation of the details of the object.

We extend the geometric error metric to a symmetric measure of quadratic error metric (SMQE). SMQE is the AMQE measured in the both direct and reverse directions, i.e. simplified to original mesh and original to simplified mesh.
SMQE is more costly than AMQE in terms of computation power but the advantage of using this error metric consists in the preservation of boundaries and sharp features.
Moreover, SMQE does not allow an island of a mesh to disappear.

For the both error metrics, we obtained simplified meshes more accurate than the simplification with other methods proposed in literature.

1.4 Overview of the dissertation

In the reminder of this thesis, we make an introduction in 3D digital objects, present their applications and the techniques which generate them (Chapter 2).
In Chapter 3 we review the prior work in the field of mesh simplification. We describe the methods used to simplifiy meshes with an emphasis on their advantages and disadvantages.
Our proposed simplification algorithm is presented in Chapter 4. We describe the metrics we use to evaluate the geometric error introduced in the simplification.
Chapter 5 contains the results obtained with our method. The performance of our algo-

rithm is evaluated. Our method is compared with other simplification methods proposed in literature.

Chapter 2

Surface Representation

This chapter begins with an overview of 3D digital objects and presents different acquisition techniques and methods to generate 3D objects.
A classification of the surface representation is made in the bellow with an emphasis on triangular meshes.

2.1 3D Digital Objects

The last twenty years brought an increasing presence of 3D digital objects in different domains such as movies, medicine, architecture, industrial design, engineering, cartography, etc.
Digital objects are usually represeted with the 3D surface of the object. This kind of objects are used in applications such as movies, aircraft simulation and industrial design. But, there are some domains such as finite element analysis where information from the interior of the objects is necessary. In this case, the volumetric representation of the object is created.
There are different methods to generate digital objects depending on the application domain (Figure 2.1).
For example, in animation films and computer games, the objects are synthetically created using Computer Aided Design (CAD) techniques.

In scientific visualisation, medical images, a data volume is a group of 2D slice images acquired by CT (computed tomography) or MRI (magnetic resonance imaging) scanner. From this volume, an isosurface is extracted using *marching cubes* algorithms [Lorensen

and Cline, 1987](Figure 1.3).

In computer vision, 3D laser scanners capture points of a real object and the 3D surface of the object is reconstructed from these points. For example, the Stanford's 3D model of Michelangelo's statue of David contains about one billion polygons (see Michelangelo Project). Among these points, some are redundant and could be removed without decreasing the quality of the model.

In remote sensing, range data sets are obtained using satellite photographs. With these data, the 3D reconstruction of some dangerous and inaccessible areas is realised and used for geo-exploration.

In virtual reality, digital objects are used for aircraft, spacecraft, automobile simulator or for surgery simulator, heritage and archaeology reconstruction. They are synthetically created using CAGD techniques.

In mechanical engineering, digital objects are used for structural analysis of bridges or for simulation of electromagnetic fields.
Moreover, they are used in cartography, in architecture and industrial design, in urban modeling.

As we have seen, all of the acquisition techniques presented above are capable to produce 3D models with millions of even billions of elements (vertices, edges or faces) which make it difficult to store, transmit over the internet, to render or analyse the models.
To improve these performances, simplification is necessary.
There are certain applications in which the real time rendering and the interactivity with the objects is more important than a detailed representation. This is the case with computer games or aircraft and military flight simulators. In this kind of applications, the 3D object can be simplified to multiple *level of details* (LOD).
In cartography, when a map with large scale is produced there are some details which are not necessary. For example, the rivers or roads could be represented with less details. In this manner, the storage space and rendering time are reduced.
3D objects obtained by *marching cubes* algorithm or from 3D scanner contain redundant data points. The redundant data could be removed to improve the storage memory. In animation films and computer games, the objects which are further away from the viewpoint or are small can be represented with fewer details in order to improve rendering time.

2.2 Surface Representation vs. Volumetric Representation

In computer graphics, the objects modeling can be realised using surface representation or volumetric representation (Figure 2.2).

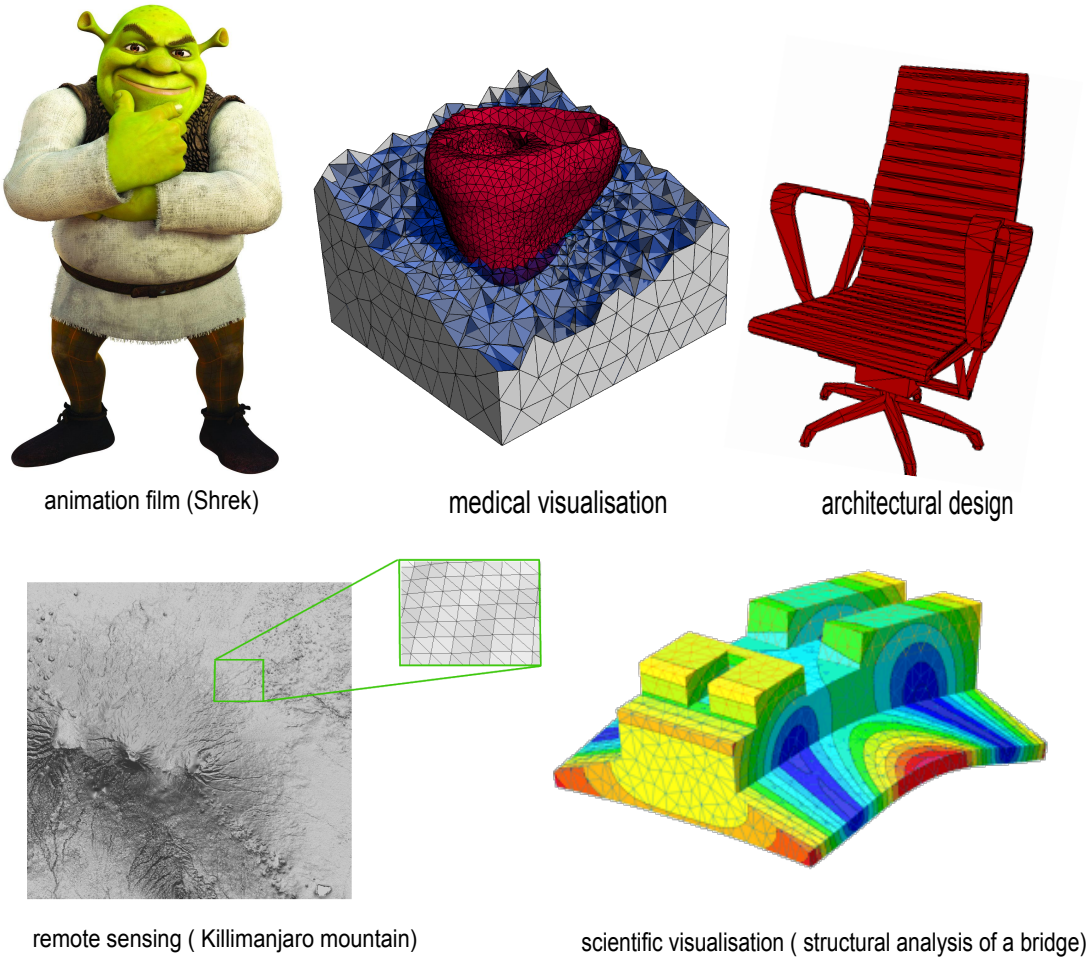


Figure 2.1: Applications of 3D digital objects in different domains. Top left: An animation film character (taken from the film Shrek). Top middle: 3D reconstruction of a heart, used in medical visualisation. The image is provided courtesy of Julien Dardenne. Top rigt: 3D surface used in architectural design. Graphisoft Desk Chair model is provided courtesy of Graphisoft by the AIM@SHAPE Shape Repository. Bottom right: example of surface reconstruction in remote sensing domain. Killimanjaro mountain model is provided courtesy of Disi by the AIM@SHAPE Shape Repository. Bottom left: digital object used in mechanical engineering, to analyse the resistance of a bridge.

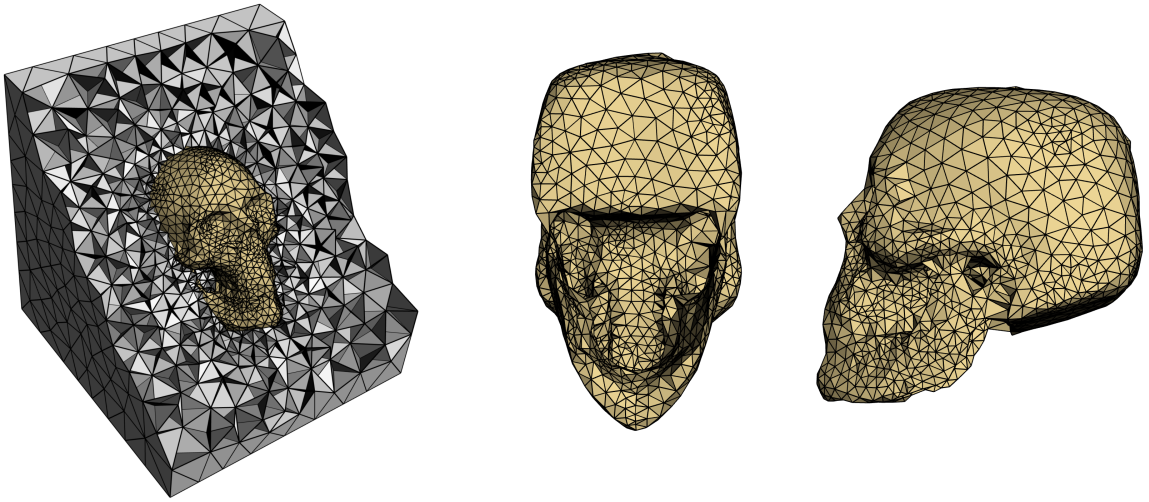


Figure 2.2: Left: Volumetric representation of a human head. Right: The surface representation. For volumetric representation more information is necessary than the surface representation. Volumetric Head model provided courtesy of Julien Dardenne.

In surface representation, we represent a 3D solid object with a 2D surface and no information about the interior of the object while in volumetric representation the interior of the object is represented as well. For example, in Figure 2.2, the volumetric representation (left part) provides information about the internal structure of a human head.
This kind of representation is useful for applications such as *finite element analysis*, where the internal structure of the object is necessary to compute the internal stresses in the object.
Volumetric representation is difficult to model, time consuming and costly in terms of storage memory space.
If we are interested only in visualisation of the object, without information about the internal structure, surface representation can be successfully used. For example, for objects in animated films, only the exterior aspect of the objects is of interest and not the interior structure, such as muscles or fibres.

This thesis concerns only 3D surface representations and the simplification algorithm is designed to reduce the complexity only for surfaces in 3D Euclidean space.

2.3 Surface Representation

In computer graphics, we deal with the 2D surface representation of a 3D solid object. The 2D surface of a 3D solid object could be regarded as the physical delimitation between the interior and the exterior of the solid.
The methods involved in 2D surface representation are divided into two main classes: parametric representation and implicit representation. Each class has its advantages and

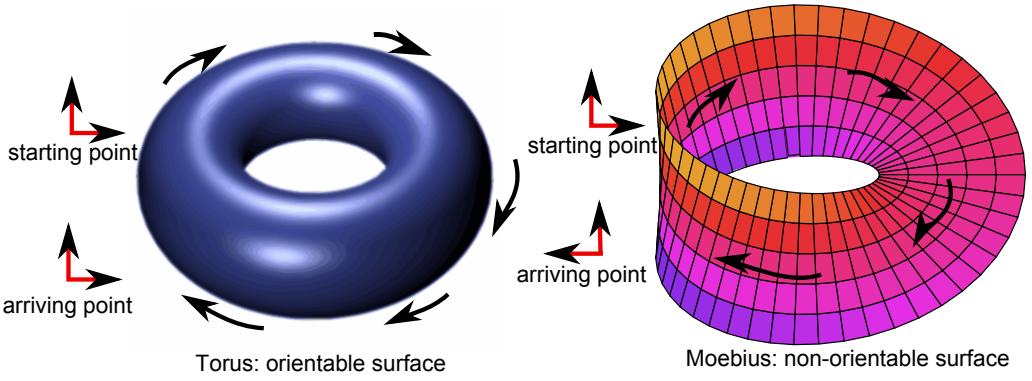


Figure 2.3: *Surface orientation. Left: Torus is an orientable surface. Right: Moebius strip is a non-orientable surface.*

drawbacks.
The parametric representation of an object is the mapping of a 2D domain through a parametric function $f : \Omega \to S$ with $\Omega \in \Re^2$ and $S \in \Re^3$. The implicit representation is actually the zero-set of a function $F : \Re^3 \to \Re^3$.
In parametric surface representation, we can distinguish between polygonal surface representation (the most popular method), spline surfaces and subdivision surfaces.

In computer graphics, a surface is most commonly defined as an *orientable continuous two-dimensional manifold embedded in* R^3 [Botsch *et al.*, 2008].
A surface can be described as being *orientable* or *non-orientable* (Figure 2.3).
A surface is called orientable if we can consistently assign a clockwise rotation as we move around. Otherwise, the surface is non-orientable.
In other words, if we assign a system of coordinates to a point on a surface (Figure 2.3) and move the point around the surface, when the point arrives in the same position and the system of coordinates has the same orientation, such as Torus model in Figure 2.3, the surface is orientable.
If the system has the same orientation for y axis and invers orientation for x axis such as the Moesbius model in Figure 2.3, the surface is non-orientable.

2.3.1 Parametric Representation

Given a plane $P \subset \Re^2$ associated to the coordinate system (u, v) and a function f which translates this plane into $\Re^3$ domain:

$$f : \begin{cases} P \subset \Re^2 \to \Re^3 \\ (u, v) \to f(u, v) \end{cases}$$

The parametric surface S is the image of the function f in $\Re^3$.

$$S = \{t \in \Re^3 | \exists (u, v) \in P,\ t = f(u, v)\}$$

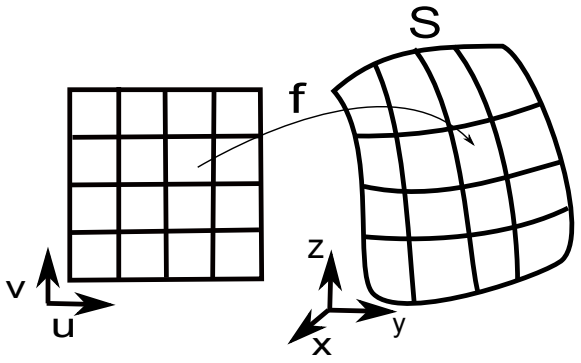


Figure 2.4: *Example of mapping from* $\Re^2$ *to* $\Re^3$.

The transformation from $\Re^2$ domain to $\Re^3$ is called *mapping*. The plane P is mapped to a surface S through a transformation given by f (Figure 2.4).
For a parametric representation it is difficult to determine the position of a point regarding a surface and moreover, it is difficult to detect self-collisions. This is one drawback of parametric representation.
Because the parametric domain and the image surface have the same topology, when the topology of the surface is changed parametrization should be updated. Thus, generation of the parametrization function is complex. This is another drawback.
Among the advantages of parametrization surface representation can be enumerated the following: sample points on the surface can be generated sampling the parameter domain and applying the parametrization function.
The geodesic neighbourhoods are easily detected by checking their neighbourhoods in the parametric domain.

Polygonal Representation

One of the most popular methods for representing the surface of a 3D object is the approximation of the surface with a collection of polygons. This collection of polygons is called a *mesh*. To create objects as in Figure 2.5, various attributes such as color, texture or images are added.

A polygonal mesh has the following advantages in being used for an object representation: it is a piecewise linear approximation with an approximation error equal to $O(h^2)$ where h is the maximum edge length; it offers an arbitrary topology, adaptive refinement and efficient rendering.
Most hardware prefers to process a large number of simple objects rather than complex objects, and for that reason polygonal meshes are preferred.
A mesh is a collection of polygons, edges and vertices (Figure 2.6):

$$\mathcal{M} = (\mathcal{F}, \mathcal{E}, \mathcal{V}) \tag{2.1}$$

Figure 2.5: *Digital objects in computer graphics. Left: The final object is the mesh from the left image with attributes of color and texture added. Image taken from animation film* Shrek.

where

$$\mathcal{F} = \{f_1, f_2, ...f_F\} \tag{2.2}$$

is the set of faces.

$$\mathcal{V} = \{v_1, v_2, ..., v_V\}, v_i \in \Re^3 \tag{2.3}$$

represents the set of vertices and

$$\mathcal{E} = \{e_1, e_2, ..., e_E\} \tag{2.4}$$

is the set of edges.
A triangular mesh presents the following relationship between faces, edges and vertices: each face has three vertices and three edges, each edge has two vertices and one (if it is a boundary edge) or two faces (if it is an interior edge). A mesh can be characterized from a *geometric* point of view or from the point of view of *connectivity*.
The geometry of the mesh is given by the coordinates of the vertices. Each vertex $v_i \in \mathcal{V}$ has associated a set of coordinates in $\Re^3$:

$$\mathcal{P} = \{p_1, p_2, ...p_V\} \tag{2.5}$$

with

$$p_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \in \Re^3 \tag{2.6}$$

The connectivity of the mesh is given by the graph $\mathcal{M} = (\mathcal{F}, \mathcal{E}, \mathcal{V})$ (Figure 2.7).
If the faces of the mesh f_i are triangles, the mesh is called a *triangular* mesh or a *quadrilateral* mesh if f_i are quadrilaterals (Figure 2.8, right). If the quadrilaterals are planar, the mesh is called *planar*. Otherwise, it is called *non-planar*.

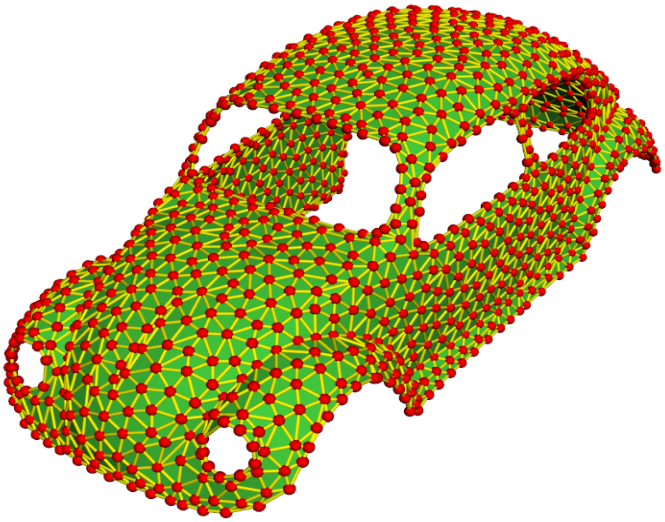


Figure 2.6: *The structure of a mesh representing a car body. The red points are the vertices, the yellow lines are the edges and green represents the faces (triangles in this example). The Beetle model is provided courtesy of L. Kobbelt.*

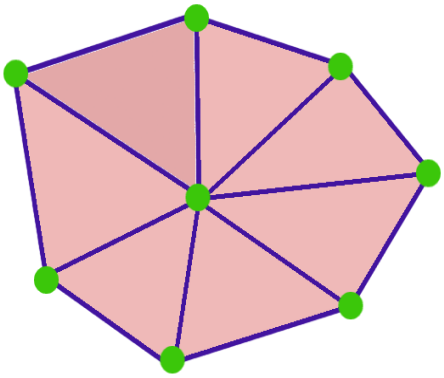


Figure 2.7: The geometry of a mesh.

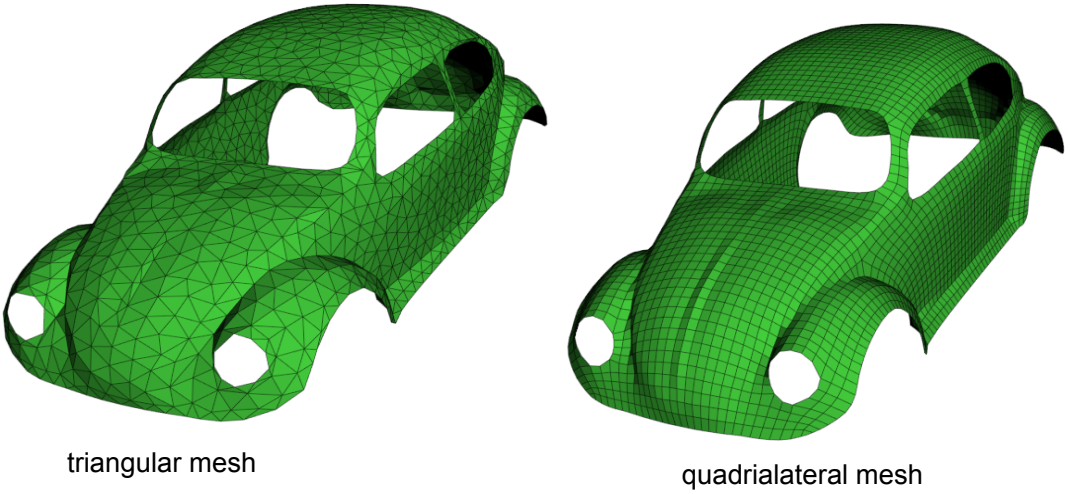


Figure 2.8: Different mesh tesselations. Beetle models are provided courtesy of L. Kobbelt.

Meshes with polygons having more than four edges are called *n-polygonal* meshes.

In computer graphics, in polygonal surface representation, the widely used primitive to approximate a surface is the *triangle*. Triangles are widely supported by graphics libraries and hardware systems and can be easily produced by 3D acquisition techniques (CT, MRI, 3D scanners, CAGD tools). For these reasons, they are the most common surface representation primitives used.
Quadrilateral meshes are used mainly for industrial and architectural design. For example, in architecture, the most utilised materials are planar ones such as glass and plywood and architectural designers prefer to work with planar quads meshes.
My thesis deals only with triangular meshes. By working with triangular meshes, we avoid the problem of unplanar faces.

Topologically, a mesh can be *manifold* or *non-manifold*.
A manifold mesh is a mesh for which the surface of every point is locally homeomorphic to a disk or a half-disk. If the surface of a point is locally homeomorphic to a disk, the manifold is called *manifold without boundaries* (the Shark in Figure 2.9). If the surface is locally homeomorphic to a semi-disk, the surface is *manifold with boundaries* (the Beetle in Figure 2.9). In other words, if an edge is shared by exactly two triangles, we deal with *manifold without boundaries* and we deal with *manifold with boundaries* if the edges are shared by two triangles and by one triangle, as well.
If an edge of the mesh is shared by more than two faces, or a vertex is shared by two disconnected fans of triangles (Figure 2.10), the mesh is called *non-manifold* mesh.
In practice, many meshes are manifold and many algorithms are designed to simplify manifold meshes, non-manifold meshes being hard to handle.
The vertex degree or valence of a vertex is the number of edges incident to that vertex (Figure 2.7). If all vertices degrees are equal, the mesh is called *regular*.
A relation between the number of faces $\mathcal{F}$, the number of vertices, $\mathcal{V}$ and the number of edges $\mathcal{E}$ of a mesh is given by the *Euler formula*:

$$\mathcal{F} - \mathcal{E} + \mathcal{V} = 2(1 - g) \tag{2.7}$$

where g is the genus of the surface (Figure 2.11). Because g is small compared to the number of elements in the mesh, the right term of the equation is almost 0. With this approximation, for a *manifold* mesh, the following approximations are considered [Botsch *et al.*, 2008]

- the number of faces is two times the number of vertices: $F \approx 2V$
- the number of edges is three times the number of vertices: $E \approx 3V$
- the number of faces around each vertex, called the valence of the vertex is on average: ≈ 6

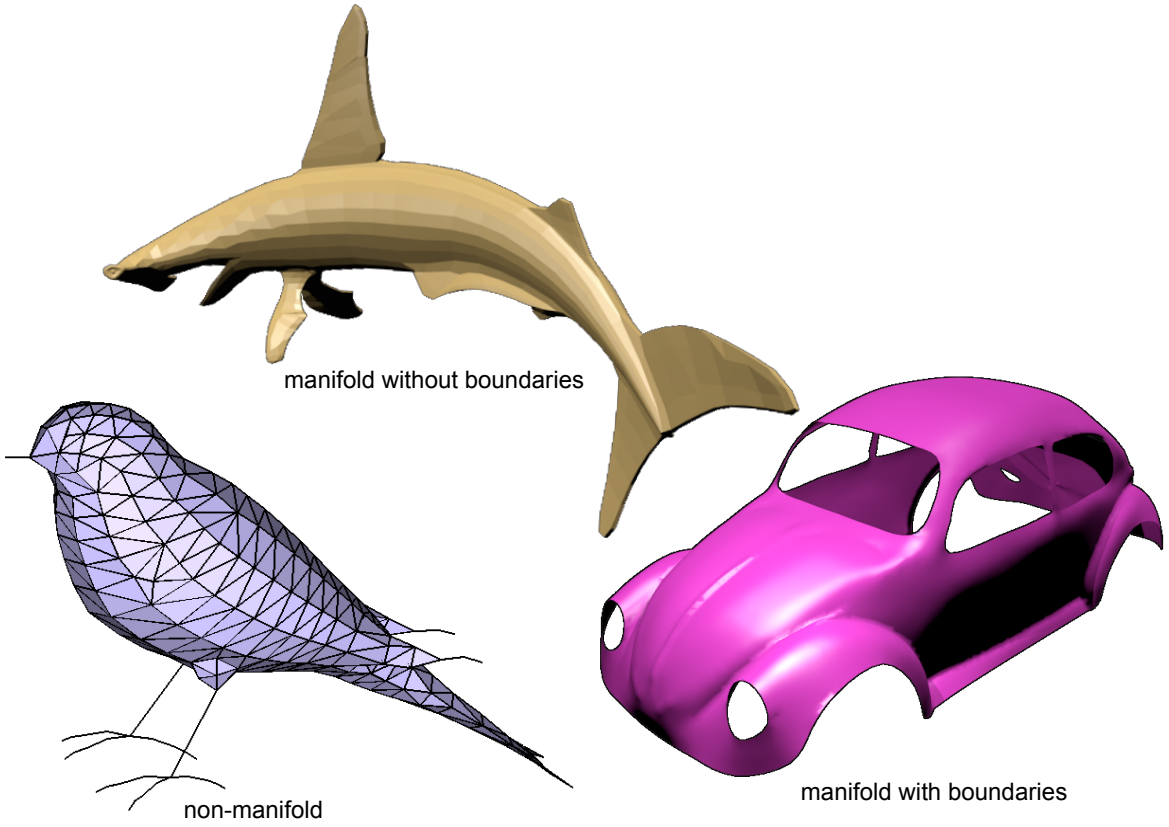


Figure 2.9: *Example of manifolds and non-manifolds meshes.* The models are provided courtesy of AIM@Shape Repository.

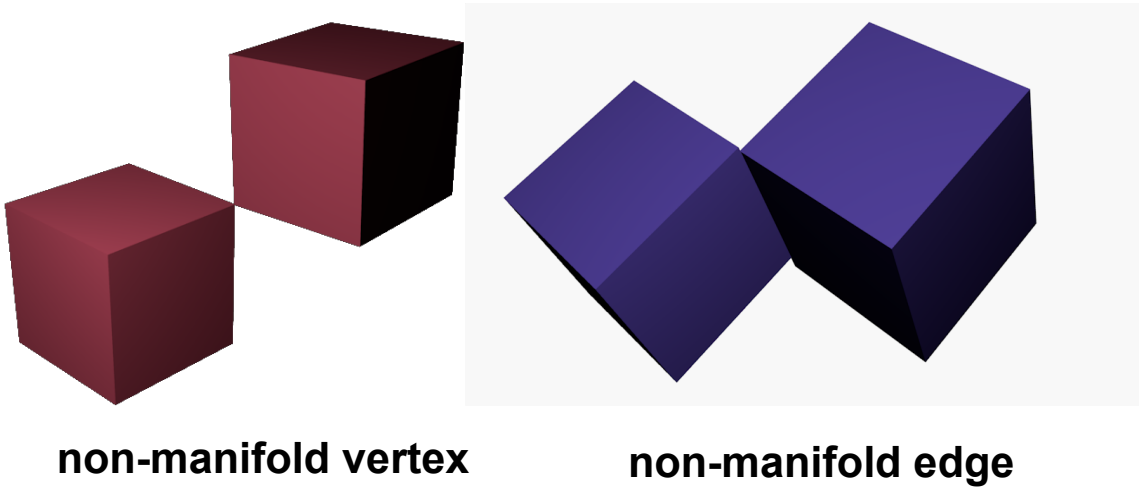


Figure 2.10: *Examples of non-manifolds meshes.* On the right, we have a non-manifold vertex and a non-manifold edge on the left.

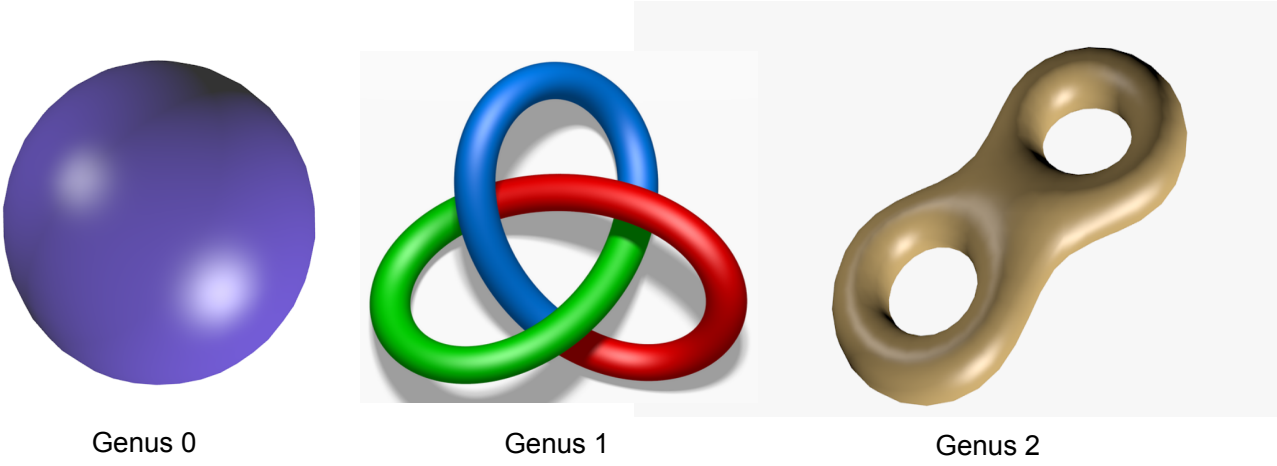


Figure 2.11: *Objects of different genus. From left to right: a sphere of genus 0, a torus of genus 1 and a double torus of genus 2.*

For a triangular mesh, each point p inside a face $f = (v_1, v_2, v_3)$ can be expressed using the face's vertices:

$$p = \alpha v_1 + \beta v_2 + \gamma v_3$$

with

$$\alpha + \beta + \gamma = 1$$
$$\alpha, \beta, \gamma \geq 0$$

α, β, γ are called *barycentric coordinates*.

Spline Surfaces

An alternative to polygonal surface representation is representing surfaces using spline functions (Figure 2.12). Spline surfaces are generated using B-spline basis functions and are the standard surface representation in Computer-Aided Design (CAD) systems.
Spline surfaces are created by mapping the domain $[0,1]^2$ into $\Re^3$ using a parametrization function

$$f : [0,1]^2 \rightarrow \Re^3$$
$$f(u,v) = \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} N_i^n(u) N_j^n(v)$$

where $N_i^n(\cdot)$ is the B-spline basis function with

$$N_i^n(\cdot) > 0$$
$$\sum_i N_i^n = 1$$

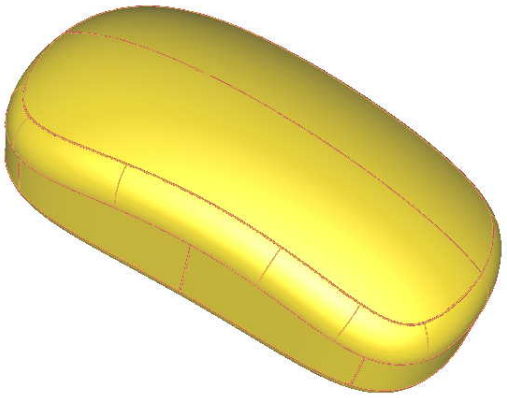


Figure 2.12: *Mouse created using spline representation.* The **Mouse** model is provided courtesy of Stephane Guillet by the AIM@SHAPE Shape Repository.

and c_{ij} are called control points. The totality of the control points determines the control mesh for the spline surface.
Each point of the surface $f(u, v)$ can be expressed as a convex combination of the control points.
The image of the grid $[0, 1]^2$ is transformed by the function f in a grid in $\Re^3$. Because of this topological constraint, to create a more complex surface using B-spline basis functions, more small patches are created and put together in a smooth manner.
A detailed presentation of spline surfaces can be found in [Farin, 2002].
Spline representations are used to create high-quality surfaces. But this representation offers some topological and geometric constraints. To represents complex topological surfaces by splines, the model should be divided into patches. After that the patches should be connected in a smooth way to obtain high-quality objects.

Subdivision Surfaces

Subdivision surfaces is another method for creating a surface starting with a control mesh (which is the coarsest one) and iteratively subdividing each face (Figure 2.13). The process of subdivision can be seen as a refinement of the mesh or smoothing of the surface. For each step of subdivision, the connectivity of the mesh is changed by subdivision of each face.
Furthermore, the geometry of the mesh is altered because after each subdivision the positions of the new vertices and sometimes the positions of the old one are recomputed.
To subdivide a face there are various methods such as Doo-Sabin, Catmull-Clark, Loop and Butterfly which propose different schemes for subdividing a face.
The geometry can be partially changed, a situation in which only the positions of new vertices should be computed or can be globally changed, in which case the old vertices change their positions, as well.
An inconvenience of the mesh subdivision is the difficulty in previewing the final smooth

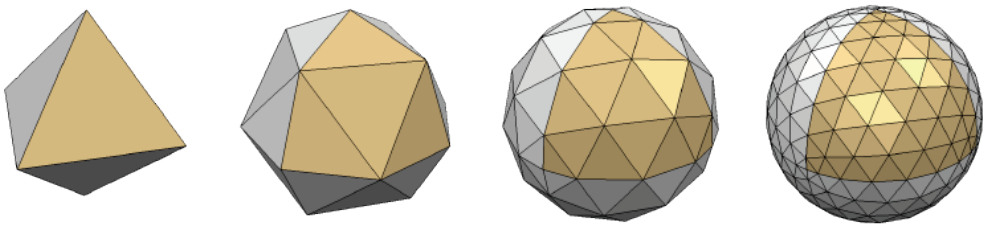


Figure 2.13: *Surface subdivision.* Each subdivision step increases the number of faces by 4. At each subdivision step, the edges are halved. Image taken from [Botsch *et al.*, 2006]

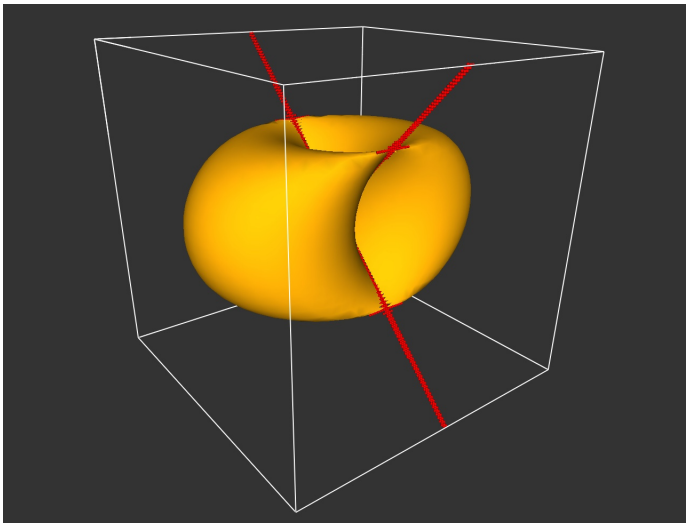


Figure 2.14: *Implicit surface representation. The* ***Bohemian Dome*** *model is provided courtesy of INRIA by the AIM@SHAPE Shape Repository.*

mesh. A coarse mesh can lead to different refinement meshes as output because of the different subdivisions applied. And this can be inconvenient for a designer.
There are, of course, advantages to using subdivision in the generation of the surfaces. One advantage is that subdivision can create surfaces with arbitrary topology. A subdivision representation permits level-of-detail rendering [Zorin and Schröder, 2000].

Differing from spline representation, subdivision representation offers more topological and geometric freedom.

2.3.2 ImplicitRepresentation

In implicit representation, the whole surface is characterized by a function. Applying this function, each point of the embedding space $\Re^3$ is classified as being inside, outside or on the surface which delimits the solid object. In this kind of representations, the surface of a solid object is considered to be the zero-level set of function

$$F : \Re^3 \rightarrow \Re$$

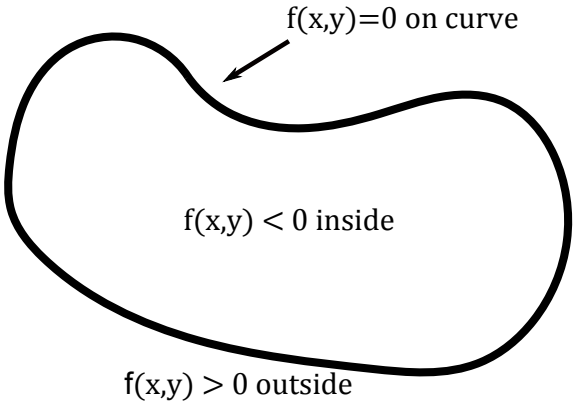


Figure 2.15: *The function evaluation for an implicit representation of a surface. If the function is positive, the point (x,y) is outside the object, if it is negative, the point is inside, and the point is situated on the object surface if the function is 0.*

Thus, each point $(x, y) \in \Re^3$ is classified as belonging to the inside, outside or on the surface of the solid, as follows (2.15):

- if F(x,y)<0 then the point (x,y) is inside the object
- if F(x,y)=0 then the point (x,y) is on the boundary of the object
- if F(x,y)>0 then the point (x,y) is outside the boundary of the object

Implicit surfaces can be seen as the level-sets of a potential function and can be successfully applied in constructive solid geometry (CSG).
The implicit surface representation has some drawbacks such as difficulty in rendering this kind of surface.
The position of the points on the surface is not known directly and we can know only if the points are inside or outside of the objects. Given a point on the surface, we do not explicitly know its neighbours.
Furtheremore, using implicit functions does not provide any parametrization of the surface.
In order to get a parametrization of the surface, a bounding box must be placed around the object. The bounding box is subdivided into a *regular grid*. The parametrized representation of the surface is done with the help of the grid's nodes. If n_i is a node of the grid, then $F(n_i)$ will be a point of the parametrization of the implicit surface.
Because for regular grid parametrization memory consumption is very large, an *adaptive structure* [Samet, 1990], [Frisken *et al.*, 2000], [Wu and Kobbelt, 2003a] can resolve this problem. The sampled values are stored in a hierarchical octree and in this manner memory is saved.

Implicit representation can be used to represent surfaces with changing topology.

Chapter 3

State of the Art of Triangular Mesh Simplification

The topic of polygonal mesh simplification is more than twenty years old, having its roots in 1976 when James Clark first introduced the term *level of details* (LOD) [Clark, 1976]. He proposed using the LOD in the representation of a scene.
From 1976 until today many polygonal simplification algorithms have been proposed.
All these simplification algorithms have the same goal: to reduce the number of polygons of the mesh but preserve the fidelity of the original. As we have seen above, the most used surface representation in computer graphics is the polygonal mesh and more exactly the triangular ones.
To capture better the details of an object, millions, even billions of triangles are used.
Almost all 3D acquisition techniques produce polygonal meshes with a lot of information and sometimes this information is redundant. Fot this reason, the meshes are simplified, but the details and characteristics of the original should be preserved as much as possible.

In mesh simplification algorithms, two main aspects are taken into consideration: the operation used to simplify the mesh and the error metric used to measure the geometric error introduced by these operations.

3.0.3 Hausdorff Distance

The *Hausdorff distance* is a measure of the distance between two subsets of the metric space.

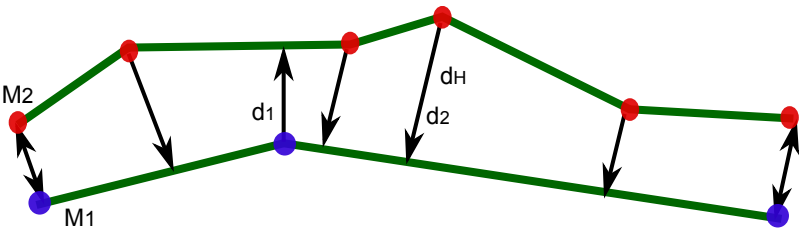


Figure 3.1: *Surface to Surface Distance*

We call *one-sided Hausdorff distance* the distance between the surfaces $\mathcal{M}_1$ and $\mathcal{M}_2$:

$$d_{H_1}(\mathcal{M}_1, \mathcal{M}_2) = max_{p \in \mathcal{M}_1} min_{q \in \mathcal{M}_2} \|p - q\| \tag{3.1}$$

where $\|\cdot\|$ is the Euclidean norm in $\Re^3$.
Since the one-sided Hausdorff distance is non-symmetric, the Hausdorff distance is defined as:

$$d_H(\mathcal{M}_1, \mathcal{M}_2) = max\{d_{H_1}(\mathcal{M}_1, \mathcal{M}_2), d_{H_2}(\mathcal{M}_2, \mathcal{M}_1)\} \tag{3.2}$$

The Hausdorff distance is the maximum of the two one-sided Hausdorff distances.

One application of the Hausdorff distance is to compare the deviation between two meshes. For example, if we have two surfaces, $\mathcal{M}_1$ and $\mathcal{M}_2$ as in Figure 3.1, the $d_{H_1}(\mathcal{M}_1, \mathcal{M}_2) = d_1$ and $d_{H_2}(\mathcal{M}_2, \mathcal{M}_1) = d_2$.
Thus, the Hausdorff distance between $\mathcal{M}_1$ and $\mathcal{M}_2$ is $d_H(\mathcal{M}_1, \mathcal{M}_2) = max(d_1, d_2) = d_2$.
Because the Hausdorff distance is an efficient tool to measure the deviation between two surfaces, it has receveid a lot of attention in recent years [Bartoň *et al.*, 2010], [Straub, 2007], [Tang *et al.*, 2009].

3.0.4 Quadratic Error

Another useful tool for measuring the distance between two surfaces is the quadratic error. The Quadratic error measures the avearage of squared distances between two surfaces:

$$E(\mathcal{M}_1, \mathcal{M}_2) = \frac{1}{a_1 + a_2} \left(\int_{p \in \mathcal{M}_1} d^2(p, \mathcal{M}_2) dp + \int_{q \in \mathcal{M}_2} d^2(q, \mathcal{M}_1) dq \right) \tag{3.3}$$

where a_1 and a_2 are the surface areas of $\mathcal{M}_1$ and $\mathcal{M}_2$.

The quadratic error measures the average squared distance between two surfaces and corresponds to the L_2 norm in the 2D case.

3.1 Metrics in practice for Mesh Simplification

In the simplification algorithms, the decision of simplification is made on an error metric criterion. Thus, the error metric used to measure the geometric deviation between the

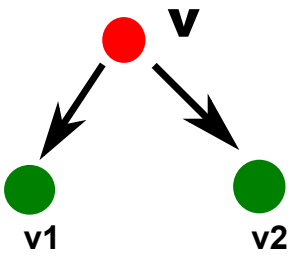


Figure 3.2: *Vertex to Vertex Distance*

simplified mesh and the original one is very important in the quality of the simplification. Bellow we present some relevant simplification algorithms from the error metrics point of view.

3.1.1 Vertex-to-Vertex Distance

One of the papers which addresses the vertex-to-vertex error metric (Figure 3.2) is [Luebke and Erikson, 1997], [Luebke, 1996].
The error introduced by collapsing vertices in a node (see Section 3.2.2) is considered to be the maximum distance between the vertices in the same cluster.
[Rossignac and Borrell, 1992] adds to the maximum vertex-to-vertex distance the inverse of the maximum angle between all pairs of edges incident to the candidate vertex. The second term is added to preserve the candidates which are situated on the object's silhouettes.

3.1.2 Vertex-to-Plane Distance

In [Schroeder *et al.*, 1992], the geometric deviation is measured as the distance from a vertex to the average plane through the vertices around that vertex. The geometric deviation is computed to the simplified mesh and not to the original one. This leads to underestimating the real error, because there is no information about the geometric error

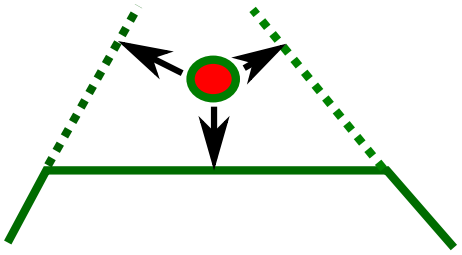


Figure 3.3: *Vertex to Plane Distance*

between the simplified mesh and original one.

In [Ronfard *et al.*, 1996] the used error metric is based on the distance between a vertex and a plane.
In this approach, each vertex of a mesh is considered as the solution of the system of equations of the supporting planes of the triangles which surround that vertex.
When an edge is collapsed to a vertex (see 3.2.1), the resulting vertex inherits all the faces inherited by the endpoints of the collapsed edge.
The error metric is the maximum distance from the resulting vertex to the suppporting planes of the faces inherited by the vertex.
The algorithm using this error metric measuring the geometric deviation produces very fast results, because of the simple computations.
The complexity of algorithm is proved to be $O(n \log n)$ [Heckbert and Garland, 1995].
Because the distance is computed between a vertex and the supporting planes of the faces of the original mesh and not directly to the faces it results in underestimating the geometric error, especially for curved surfaces.

Inspired from [Ronfard *et al.*, 1996], [Garland and Heckbert, 1997] proposes a new error metric to approximate the geometric deviation. Their error metric is the sum of the squared distances from a vertex to the supporting planes of faces inherited by the vertex.
For a face of the mesh, the equation of its supporting plane can be written as:

$$n^T v + d = 0 \tag{3.4}$$

where n is the unit normal of the plane, v is a vertex belonging to the face and d is the displacement of the plane.
If a vertex does not belong to the plane, then the value $n^T v + d$ is the distance from the vertex to the plane.
Thus, the squared distance is

$$d^2 = (n^T v + d)^2 = v^T n n^T v + 2 d n^T v + d^2 \tag{3.5}$$

Eqn. 3.5 can be written as:

$$d^2 = v^T A v + 2 b^T v + c \tag{3.6}$$

with

$$A = n n^T \tag{3.7}$$

We denote by

$$Q = (A, b, c) \tag{3.8}$$

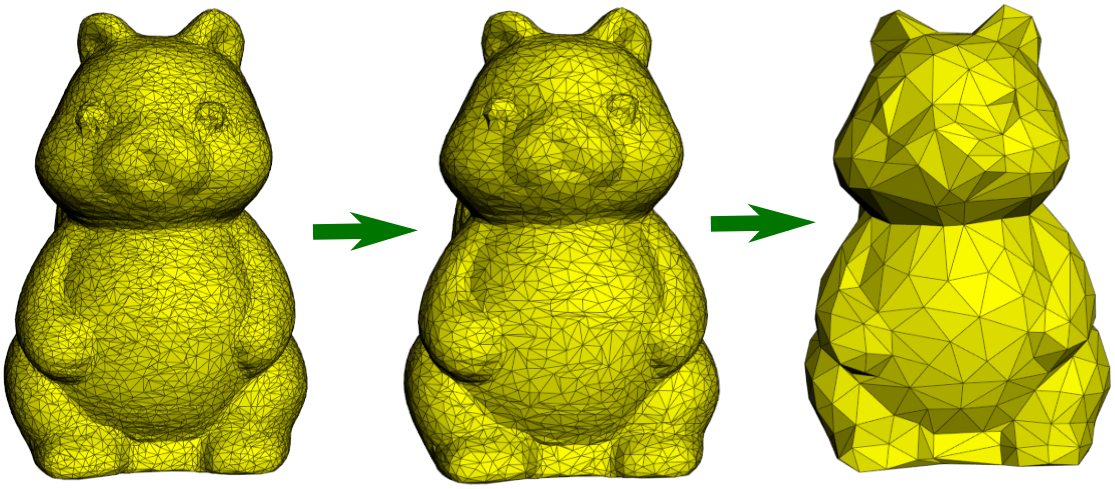


Figure 3.4: *Sequence of simplifications with QEM.* The right model has 9995 vertices. The simplified versions have 5000, respectively 500 vertices. Squirrel model provided courtesy of MPII by the AIM@SHAPE Shape Repository.

the symmetric quadratic matrix associated to a plane defined by (n, d).
Based on the observation that *quadrics* are additive, [Garland and Heckbert, 1997] compute the squared distance from a vertex to the supporting planes of the faces inherited by the vertex as:

$$E(v) = v^T(\sum_i Q_i)v \tag{3.9}$$

where $\sum_i Q_i$ is the sum of quadrics of supporting planes for the faces inherited by the vertex.
Thus, for a pair contraction (see 3.2.1), the quadric associated to the resulting vertex is the sum of the quadrics of each vertex which forms the pair.
In this context, the position of the resulting vertex is the one which minimizes the quadratic error associated to that vertex. Thus, the optimal position is the solution of the system:

$$\mathbf{Ax} = \mathbf{b} \tag{3.10}$$

$$x = -A^{-1}b \tag{3.11}$$

It is assumed that A is invertible and well-conditioned.
In this method as in the method proposed in [Ronfard *et al.*, 1996], the geometric deviation is computed as the distance from a vertex to the supporting planes of the triangles inherited by the vertex and not directly to the triangles. This leads to a rough estimation of the error.
Because the computation error is reduced to some additive and multiplicative operations, the algorithm is very fast.

[Erikson and Manocha, 1999] propose a simplification algorithm based on the quadratic error metric, but their algorithm brings some improvements. First, they introduce an adaptive and automatic threshold in the selection of vertices pairs (see Section 3.2.1). In the algorithm proposed in [Garland and Heckbert, 1997], a fixed user defined threshold is used for all simplifications. If the threshold is too small, too many pairs of vertices will be considered as candidates and the complexity of the algorithm increases. If the threshold is too big, many pairs of vertices are not taken into consideration as candidates and the quality of the results is not satisfactory.
An adaptive threshold can resolve this problem. The algorithm proposed in [Erikson and Manocha, 1999] starts with a small threshold and this is automatically increased during simplification.
The second contribution is a triangle-area weighted quadratic error. The error associated to each vertex is the sum of trianle-area weighted quadric matrices of the supporting planes of the triangles inherited by the vertex divided by the sum of triangles-areas. Because the quadratic error is multiplied by the area of the triangle, the simplification algorithm becomes more robust to the mesh tessellation. And that improves the fidelity for some models, such as meshes with skinny triangles.
In addition, in this algorithm, the authors take into account the simplification of mesh attributes such as: color, normal and texture coordinates. The algorithm is able to simplify the topology and geometry of non-manifold meshes producing high-quality results in a fast running-time.

[Garland and Heckbert, 1997] extends to meshes with material properties such as: colors, texture, normals [Garland and Heckbert, 1998]. To handle the mesh attributes in the simplification algorithm, they introduce the attributes coordinates in quadrics.
The attributes of a mesh are addressed in the paper [Hoppe, 1999].

3.1.3 Surface-to-Surface Distance

A more precise but more expensive method than those presented above is the approximation of the geometric deviation based on the estimation of the Hausdorff distance [Klein *et al.*, 1996]. In this algorithm, the geometric error is characterized using the Hausdorff distance between the simplified mesh and the original one.
An exact Hausdorff distance is computed, the geometric deviation being accurately measured with the drawback of being more time consuming.
This approach is different from the approaches presented above, where no information about the global deviation of the final simplified mesh from the original one is provided.

To reduce the computational complexity of the error, [Ciampalini *et al.*, 1997] proposes an one-sided Hausdorff distance approach with a number of heuristics. It supposes that

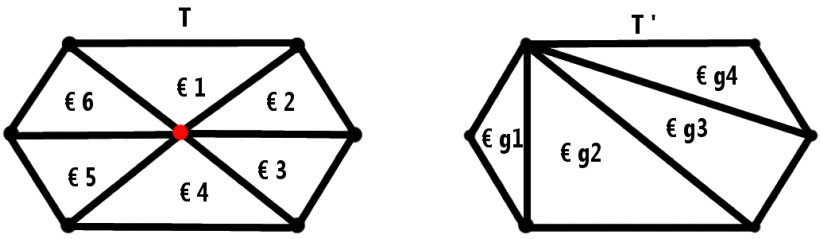


Figure 3.5: Representation of a triangulation and its corresponding simplified version.

there are two triangulations (Figure 3.5) T the corresponding triangulation on T after simplification.
For each face of the original triangulation, T has an associated error, denoted in Figure 3.5 by ϵ_i, $i = 1 : 6$. The global error of a face f in the new triangulation T' is computed as:

$$\epsilon_{g1}(f) = max_{p \in P} dist(p, T) + max_{i=\{1..6\}} \epsilon_i \tag{3.12}$$

where P is the set of sample points on the face f and ϵ.
Thus, the error for a face of the simplified triangulation, T' is the sum of the maximum distance between the previous triangulation, T and the origianl mesh and the maximum deviation between the new triangulation, T' and T.
Because $\epsilon_{g1}(f)$ provides an over-estimation of the real error, the authors introduced the second term to compensate this over-estimation.
For the second term of the error approximation, each face in the simplified mesh has an associated list of removed vertices from the original mesh for each of the faces which is closest. When a vertex v is removed, v is attributed to the closest face from the new triangulation T'. Thus, all the removed vertices attributed to the faces in T are redistributed to the new faces from T'.
$\epsilon_{g2}(f)$ is considered to be:

$$\epsilon_{g2}(f) = max_{v_i \in P} dist(v_i, f) \tag{3.13}$$

where P is the list of redistributed vertices.
Second term of the global error produces an under-estimation of the actual error.
Thus, the geometric deviation associated with a face, f in the simplified mesh is defined as:

$$\Upsilon(f) = \epsilon_{g1}(f) + \epsilon_{g2}(f) \tag{3.14}$$

Both of the algorithms presented above take into consideration the global error between the simplified mesh and the global one.
There are certain applications such as organs 3D reconstruction from CT data in order to create a prosthesis [Klein *et al.*, 1996] where knowing the global geometric deviation is

important. Most simplification algorithms take into account only the geometric deviation between the simplified mesh and the original to describe the quality of the simplification [Cignoni *et al.*, 1996].

If the goal of the simplification is to speed up rendering, a measure of the quality of the simplification is the perceptual quality. The papers which address this measure of quality are [Lindstrom and Turk, 2000], [Cohen *et al.*, 1998].

[Cignoni *et al.*, 1996] compute the Hausdorff distance in order to evaluate the geometric deviation between two meshes. For a more accuracy in the evaluation of the deviation, the faces of the meshes are sampled. Thus, the Hausdorff distance is computed between the sample points. This algorithm is an efficient tool to measure the deviation between two meshes, known in literature as *Metro*. The drawback of *Metro* is that it cannot handle very large meshes.
To resolve this drawback, [Meftah *et al.*, 2010] proposes a method computing the distance between two meshes on the fly.
The geometric deviation between two meshes is evaluated using the root mean square error (*RMSE*) computed in the both, direct and backward directions. The largest between the direct and backward distances is considered to be the geometric deviation between the meshes. To compute on the fly the RMSE between the meshes a more simplified version of one of the meshes is used.A correlation between the regions of the two meshes is made through the faces of the simplified mesh. Thus, at a time, the distances are computed only between the corresponding regions.
Using an out-of-core algorithm, the distance can be evaluated even for huge meshes.

Simplification Envelopes

In [Cohen *et al.*, 1996] two envelopes are built to control the mesh simplification based on the Hausdorff distance.
The method generates an outer and an inner envelope which bound the mesh. The envelops guarantee that the deviation of the simplified mesh is no higher than a predefined threshold, ϵ.
To build the outer envelope, each vertex in the original mesh is moved with ϵ on the normal direction. The position for a vertex v' on the outer envelope is:

$$v' = v + n\epsilon$$

where n is the normal of the vertex v.
For the inner envelope, the vertices are displaced by $-\epsilon$ than their original positions.
When vertices from curved regions are displaced, self-intersections can appear on the envelopes. To avoid this, ϵ is decreased.
Using the envelopes guarantees that the simplified mesh is closer to the original than a

predefined tollerance, that topology is preserved and the simplified mesh does not have self-intersections.
This method requires a manifold input in order to construct the envelops. In addition, because of strict conditions regarding topology preservation and self-intersection avoidance, sometimes, high rates of simplification cannot be obtained.

Another approach based on envelopes is proposed in [Borouchaki and Frey, 2005]. A set of two envelopes situated at a predefined δ Hausdorff distance from the original mesh are created. During mesh simplification, the geometric deviation should remain inside the envelopes. The envelopes are used to ensure the geometric constraint.
Together with the geometric constraint, a smoothness constraint is also used. To ensure the smoothness of the simplification, for each modified triangle, the product between the normal vector of the triangle and the normal vector to a vertex of the triangle should be lesser than a predefined value.
A simplification operator is considered to be a candidate if, for each triangle modified with the application of this operator the following requirements are fullfiled: the Hausdorff distance between the triangle and the original mesh does not exceed δ and the deviation between the normal of the triangle and the normal of the vertex does not exceed a predefined value.
Because, computing the exact Hausdorff distance between a triangle and the original mesh is time consuming, the following approximations are made: being $\mathcal{T}$ the set of triangles before simplification and $\mathcal{T}'$ the set of simplified triangles corresponding to $\mathcal{T}$.
For a triangle $t' \in \mathcal{T}'$, the Hausdorff distance to the original mesh is approximated with the sum of the Hausdorff distance between t' and the set of triangles before simplification, $\mathcal{T}$ and the Hausdorff distance between $\mathcal{T}$ and original mesh, $\mathcal{M}$as:

$$d_H(t', \mathcal{M}) \leq d_H(t', \mathcal{T}) + d_H(\mathcal{T}, \mathcal{M})$$

which can be written as:

$$d_H(t', \mathcal{M}) \leq d_H(t', \mathcal{T}) + max_{t \in \mathcal{T}} d_H(t, \mathcal{M})$$

so

$$d_H(t', \mathcal{T}) + max_{t \in \mathcal{T}} d_H(t, \mathcal{M}) \leq \delta$$

Only the distance $d_H(t', \mathcal{T})$ should be computed in order to determine the upper limit of the Hausdorff distance between a triangle and the original mesh.
The one-sided Hausdorff distance from a triangle $t \in \mathcal{T}$ to the region $\mathcal{T}'$ is approximated as follows: t is projected onto the plane of $t' \in \mathcal{T}'$. We denote the projected triangle with $t_{\mathcal{T}'}$. The intersection of t' with $t_{\mathcal{T}'}$ is a polygon noted $\mathcal{P}_{\mathcal{T}'}(t)$:

$$t' \cap t_{\mathcal{T}'} = \mathcal{P}_{\mathcal{T}'}(t)$$

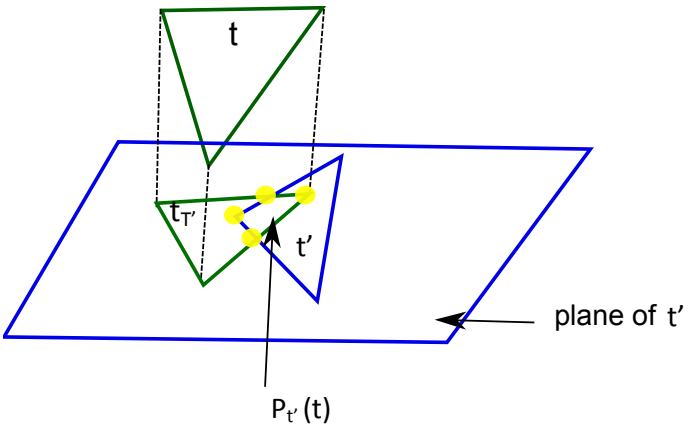


Figure 3.6: *Example of the approximation of the Hausdorff distance between two triangles.S*

It is supposed that the projection of $\mathcal{T}$ onto $\mathcal{T}'$ is restricted to $\mathcal{T}'$, that the region $\mathcal{T}'$ can be written as:

$$\mathcal{T}' = \cup_{t \in \mathcal{T}} t_{\mathcal{T}'}$$

and each triangle in $\mathcal{T}'$ is:

$$t' = \cup_{t \in \mathcal{T}} \mathcal{P}_{\mathcal{T}'}(t)$$

The one-sided Hausdorff distance between a triangle $t' \in \mathcal{T}'$ and $\mathcal{T}$ is approximated by the maximum Euclidean distance between the vertices of the polygon $\mathcal{P}_{\mathcal{T}'}(t)$ and the vertices of the projection of this polygon on the plane of triangle $t \in \mathcal{T}$.
Thus, the Hausdorff distance between a triangle t' and a region $\mathcal{T}$ is approximated to:

$$d_H(t', \mathcal{T}) = max_{t \in \mathcal{T}} \delta(t, t')$$

where

$$\delta(t, t') = max(d(\mathcal{P}_{\mathcal{T}'}(t), \mathcal{P}_{\mathcal{T}'}(t)^{-1}), d(\mathcal{P}_{\mathcal{T}}(t'), \mathcal{P}_{\mathcal{T}}(t')^{-1}))$$

where $\mathcal{P}_{\mathcal{T}'}(t)^{-1}$ is the projection of $\mathcal{P}_{\mathcal{T}'}(t)$ onto the plane of t.
By providing a set of two envelopes, the deviation of the simplified mesh is guaranteed to be within this deviation. Moreover, by using smoothness constraints, the quality of the simplified mesh is controlled.

Permission Grids

To control the geometric deviation of simplification, [Zelinka, 2002] proposes a method called *permission grid*. A 3D grid is placed around the object. The faces of a mesh are allocated of the grid's cells. At the beginning, all 3D cells which are closer than a tolerance ϵ to the original mesh have the same flag. If not, the cells are set to *empty*. During simplification, if the simplification operation generates a triangle which intersects an empty cell, the operation is disallowed.
This method simplifies the topology of the model as well. Being different from existing

error-bounds methods, this algorithm takes as input arbitrary triangular meshes. The error tolerance of the grid can be expanded during simplification and this makes possible the level of detail representation. The size of the permission grid is independent from the size of the input mesh. But the size is dependent on the desired tolerance. For smaller tolerances, larger grids are required.
An important aspect of the permission grid is the resolution.
There are some classes of inputs for which the permission grid algorithm is not advantageous. One of these is the class of inputs which are disjointed into different pieces.

A simplification algorithm which is able to reduce both the geometry and the scalar variables attached at each mesh's vertex while guaranteeing the limit of error is proposed in [Bajaj, 1996]. This simplification algorithm preserves the mesh's topology and takes as input an arbitrary mesh.

[Hoppe *et al.*, 1993] proposes a global energy function to control the simplification. This function includes three terms: a term which controls the number of vertices in the approximated mesh, a term which controls the fidelity of the simplified mesh regarding the original one and a spring term which penalizes the long edge in the triangulation. During the simplification, the energy function is minimized.
A user-specified number of points is spread over the original mesh and gradually the points are moved in order to minimize the energy function introduced above. An important remark is that the set of points initially spread over the mesh has to contain fewer vertices than the original mesh.
When these points are placed in positions for which the energy function is minimum, a triangulation is applied over the points. In this manner, a simplified mesh which is the close to the original one in terms of the energy function is constructed.
The vertices of the simplified mesh are not a subset of vertices of the original mesh. This can be considered an advantage, because the simplification is not sensitive to the noise in the input.

Re-tilling Method

A *re-tilling* method is proposed in [Turk, 1992]. A set of points is initially spread on the original mesh and moved away through a relaxation step in order to achieve an uniform distribution. After that these points are introduced in the mesh triangulation. Thus, a new triangulation of the mesh is done with the new vertices together with the old ones. Then, the original vertices are removed from the triangulation one by one. In the end, the simplified mesh contains only the new vertices.
The simplification is designed to preserve the topology of the input, but the algorithm does not preserve sharp edges. An advantage is that it works on manifold as well as on non-manifold inputs.

Most recently, [Digne *et al.*, 2012] proposed an algorithm which reconstructs a surface from a set of points through surface simplification. An initial triangulation from the input set of points is simplified.The involved error metric which classifies the candidates operators is the optimal transport from the input points to the faces of the triangulation.
To define the optimal transport, the mesh is discretized into points, called *bins* by the authors. Thereafter, a vertex is its own bin, an edge is decomposed into a user-specified number of bins and a face is decomposed by 2D Centroidal Voronoi Tessellation and the site of each Voronoi cell is considered to be a bin.
Thus, the cost of the simplification operator is quantified using the following metric:

$$cost(\pi) = sum_{ij} m_{ij} \|p_i - b_j\|^2 \tag{3.15}$$

where π is the transport plane between points in the input set p_i and points in the bins, b_j.
m_{ij} defines a set of weights with the following properties:

$$\forall ij : m_{ij} \geq 0 \tag{3.16}$$

$$\forall i : \sum_j m_{ij} = m_i \tag{3.17}$$

$$\forall j : \sum_i m_{ij} = c_j l_{s(j)} \tag{3.18}$$

l_i are additional variables used to enforce the uniformity constraints.
Once the simplification is performed, the algorithm improves the position of the remaining vertex (the vertex after the application of simplification operator). The vertex is moved to an optimal position which minimizes the cost of the transport plan, π. After the optimal position is found, the optimal transport plan is updated around the vertex.
More precisely, the optimal vertex position should minimize the following expression:

$$min_v \sum_i \sum_j m_{ij} \|p_i - \alpha_j v - \beta_j v_1 - \gamma_j v_2\|^2 \tag{3.19}$$

where v_1 and v_2 are the vertices of the triangle $t = (v_1, v_2, v_3)$, α_j, β_j and γ_j are the barycentric coordinates of bin b_j.
The optimal vertex position with respect to triangle t is then:

$$v^* = \frac{\sum_i \sum_j m_{ij} \alpha_j (p_i - \beta_j v_1 - \gamma_j v_2)}{\sum_i m_{ij} \alpha_j^2} \tag{3.20}$$

Shape Approximation

A different simplification approach is based on *proxies*. A *proxy* $\mathcal{P}r_i$ is a plane defined by its normal vector n_i and a point which belongs to the plane, x_i.

Simplification algorithms based on proxy fitting subdivide the mesh into a set of disconnected patches:

$$\mathcal{M} = \{\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_N\}$$

with

$$\mathcal{P}_1 \cap \mathcal{P}_2 ... \cap \mathcal{P}_N = 0$$

$$\mathcal{P}_1 \cup \mathcal{P}_2 ... \cup \mathcal{P}_N = \mathcal{M}$$

and each patch is approximated by a proxy:

$$\mathcal{P}r_i = (x_i, n_i)$$

The optimal shape approximation is, for a given number N of proxies, an error metric E and an input mesh $\mathcal{M}$ to find a set of patches $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_\mathcal{N}\}$ and the optimal set of proxies $\mathcal{P}r = \{\mathcal{P}r_1, \mathcal{P}r_2, ..., \mathcal{P}r_N\}$ which minimize the following error metric:

$$E(\mathcal{P}, \mathcal{P}_r) = \sum_{i=1}^{N} E(\mathcal{P}_i, \mathcal{P}r_i) \tag{3.21}$$

To minimize the error metric $E(\mathcal{P}, \mathcal{P}r)$ an approach is proposed in [Cohen-Steiner *et al.*, 2004].
Their method, called *variational shape approximation* iteratively alternates between *region partitioning* and *proxy fitting*.
In *region partitioning*, the regions are adapted to a set of given proxies in order to minimize the error $E(\mathcal{P}, \mathcal{P}r)$.
In proxy fitting, the proxies are adapted to the set of given regions, in order to minimize the error.
At the beginning, n regions are created by randomly picking n triangles form the input mesh. The set of proxies is initialised $\mathcal{P}r_i = (x_i, n_i)$ where x_i is a point belonging to the triangle t_i and n_i is the normal of the triangle t_i. The priority of the triangle t_i is $E(t_i, \mathcal{P}r_i)$. This priority is introduced to a priority queue.
For each triangle t_i, the neighbouring triangles are identified and their priorities are introduced in the priority queue.
Each triangle assigned to a region is set to *conquered*. The pairs $(t_i, \mathcal{P}r_i)$ are pop up from the queue and if they are not set to *conquered*, the triangle is assigned to the corresponded region. The algorithm is stopped when the priority queue is empty. This phase is the geometry fitting.
After the regions are created, proxy fitting phase is started.
The regions are kept fixed and the proxies are adjusted in order to minimize the global error $E(\mathcal{P}, \mathcal{P}r)$.

In [Cohen-Steiner *et al.*, 2004], the authors propose two error metrics:

$$\mathcal{L}^2(\mathcal{P}_i, \mathcal{P}r_i) = \int_{x \in \mathcal{P}_i} (n_i^T x - n_i^T x_i) dA \tag{3.22}$$

The error metric is actually the squared distance from a point x to a plane $\mathcal{P}_i$. For this metric, the best proxies are the least-square fitting planes. The authors propose computing this metric with principal component analysis.
The second error metric introduced is

$$\mathcal{L}^{2,1}(\mathcal{P}_i, \mathcal{P}r_i) = \int_{x \in \mathcal{P}_i} \|n(x) - n_i\|^2 dA \tag{3.23}$$

For this metric, the normal of a proxy is the area-weighted triangle normals which belong to that proxy.
After the optimal partitioning of a mesh is done and the corresponding proxies are identified, an anisotropic remesh is started. All vertices of the original mesh which are situated at the intersection of three or more regions are identified and projected into the corresponding proxies. The average position is computed. A Delaunay triangulation over the input mesh lead to an anisotropic remeshing.
A different approach for computing the minimum of the global error $E(\mathcal{P}, \mathcal{P}r)$ is proposed in [Marinov and Kobbelt, 2005] and it is based on a greedy algorithm.
The algorithm uses a set of patches, $\mathcal{P} = \{\mathcal{P}_1, ..., \mathcal{P}_N\}$, a set of proxies, $\mathcal{P}r = \{\mathcal{P}r_1, ..., \mathcal{P}r_N\}$ and a set of polygonal faces, $\mathcal{F} = \{\mathcal{F}_1, ..., \mathcal{F}_N\}$. Each face is an arbitrarly connected polygon.
In the beginning, a seed triangle is set to each patch: $\mathcal{P}_i = \{t_i\}$, the proxy $\mathcal{P}r_i$ is set to the plane through the triangle t_i: $\mathcal{P}r_i = (x_i, n_i)$ and $\mathcal{F}_i = \{t_i\}$.
The goal of the algorithm is to maintain the projection of $\mathcal{F}_i$ on the proxy, $\mathcal{P}r_i$ injective. With this constraint, the degenerate faces and fold-overs are avoided. Furthermore, a triangular mesh can be extracted with each iteration.
The patches are greedily adjusted until a predefined maximum error or a predefined number of patches is achieved. In each iteration two patches are merged into a new patch with the constraint of preserving the injectivity: $\mathcal{P} = \mathcal{P}_i \cup \mathcal{P}_j$. The proxy for the new patch is the area-weighted average of proxies $\mathcal{P}r_i$ and $\mathcal{P}r_j$:

$$\mathcal{P}r = (x, n)$$

where

$$n = \frac{a_i n_i + a_j n_j}{\|a_i n_i + a_j n_j\|}$$

and

$$x = \frac{a_i x_i + a_j x_j}{a_i + a_j}$$

where $a_i = area(\mathcal{P}_i)$.
A new face, $\mathcal{F}$ is computed from $\mathcal{F}_i$ and $\mathcal{F}_j$. All boundary valence-2 vertices are removed if they are closer to the corresponding proxy than a user predefined threshold.
For each two adjacent patches, the error metric is computed $E(\mathcal{P}, \mathcal{P}r)$ and inserted into a priority queue. Then, the patches which produce the minimum error are merged.
The error for the new faces is computed as:

$$L^2(\mathcal{F}) = \mathcal{L}^2(\mathcal{D}_i, \mathcal{P}r) + \mathcal{L}^2(\mathcal{D}_j, \mathcal{P}r)$$

where $\mathcal{D}_i$ and $\mathcal{D}_j$ are two sub-regions of $\mathcal{P}_i$ and $\mathcal{P}_j$ with fewer faces.
This approximation is done with the goal to speed up the algorithm.

$$L^{2,1}(\mathcal{F}) = a_i \|n_i - n\|^2 + a_j \|n_j - n\|^2$$

where $a_i = area(\mathcal{P}_i)$.
Finally, the global error for a polygonal face is:

$$E(\mathcal{F}) = (1 + L^2(\mathcal{F}))(1 + L^{2,1}(\mathcal{F}))$$

Because the mesh extraction step from [Cohen-Steiner *et al.*, 2004] can produce degenerate triangles and fold-overs, the algorithm proposed in [Marinov and Kobbelt, 2005] addresses this drawback by using an injectivity constraint. Thus, a simplification operation is allowed if the boundaries of the faces obtained after simplification are injectively projectable into a plane.
Because of the greedy character, the algorithm can get stuck in a local optimum. However, the variational shape approximation algorithm produces better approximations with a lower approximation error.

3.2 Mesh Simplification Operations

Mesh simplification operations can be divided into two classes: those which perform local modifications of the mesh and those which are successively applied in order to simplify the mesh. The second class includes the simplification operators which simplify the mesh globally. Those operators simplify the mesh in one step.
As regards those operations, we can distinguish between:

- **iterative simplification**: performs local modification on the mesh. To achieve a global mesh simplification, iterative modifications are performed. Because of the local modifications, this class of algorithms is accurate, producing simplifications with a high fidelity to the original. More so, these algorithms produce level-of-detail (LOD) representations, because of the iterative characteristic.

- **direct simplification**: performs the simplification of a mesh in one step. It can be very fast, but the quality of the results is low. These algorithms are not able to generate LOD representations.

Next, we will detail two categories of simplification algorithms, presenting their advantages and disadvantages.

3.2.1 Iterative Simplification

Algorithms based on iterative simplification are among the most used algorithms. A polygonal mesh is simplified by iteratively applying a local operator. The basic local operators are: vertex removal, face removal, edge and half-edge collapse, vertices merging or faces merging. Besides these operators, there are operators such as vertex split and edge split which are used together with the operators introduced above.
Because these operators perform local modifications, in order to generate higher rates of simplifications, the local operator is applied iteratively until the desired error is achieved. For each iteration, the candidates are classified according to the geometric deviation introduced in the mesh by their application. The candidate which produces the minimum geometric deviation is selected and applied to the mesh. The process is repeated until a user-defined condition is achieved (the target number of elements or a pre-defined error). Being an iterative process in which for each iteration small changes are applied to the mesh and the candidates are re-evaluated, the algorithms produce high-quality approximations. The weakness consists in computational complexity, which is $O(nlogn)$ up to $O(n^2)$ in the worst cases (especially when there is a global threshold) [Botsch *et al.*, 2008].
In this category of simplification algorithms two aspects should be detailed: the operator used for simplification and the error metric used to measure the geometric deviation introduced by the simplification operator.

Next, some local simplification operations are detailed.

Vertex Removal

Among the first local simplification operators used in mesh simplification is vertex removal, proposed in [Schroeder *et al.*, 1992]. This operator removes one vertex and the faces surrounding that vertex (Figure 3.7) [Ciampalini *et al.*, 1997], [Soucy and Laurendeau, 1996], [Soucy and Laurendeau, 1992], [Klein *et al.*, 1996], [El-Sana and Varshney, 1997], [Bajaj, 1996].
The resulting hole is then re-triangulated. If the valence of a vertex (the number of faces shared by the vertex) is n, for a manifold without boundaries, after a vertex removal, the number of faces becomes $n-2$. Vertex removal reduces the complexity of a mesh by one vertex and two faces.
With each iteration step, each vertex of the mesh is evaluated. The geometric deviation introduced by the vertex removal is computed. After all possible candidates are simulated,

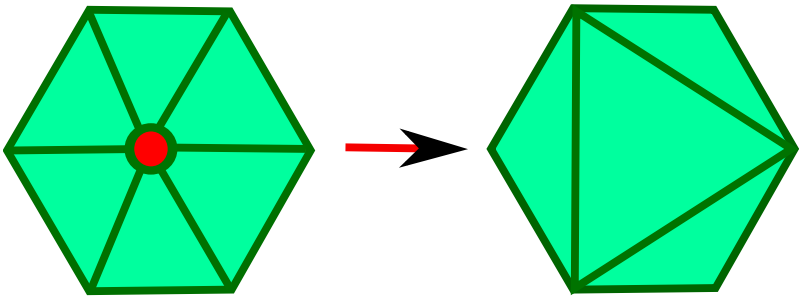


Figure 3.7: *Vertex removal. One vertex and the adjacent triangles are removed. The resulting hole is triangulated with fewer triangles.*

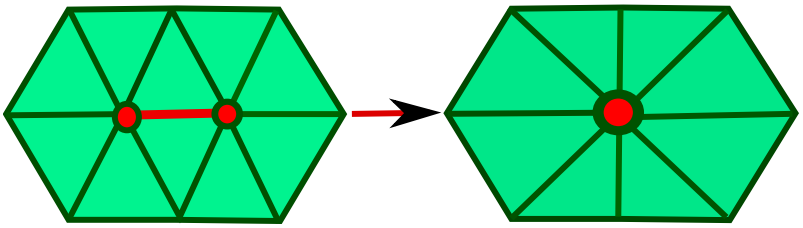


Figure 3.8: *Edge Collapse. One edge is collapse to a single vertex. Triangles which share the deleted edge are deleted, too. The number of vertices is reduced by one, the number of faces by two and the number of edges by three.*

the vertices are classified according to geometric deviation.
The vertex which introduces the least geometric deviation is chosen for removal and the remaining vertices are re-evaluated. The vertex removal operator has certain advantages such as simplicity and speed. The complexity of the algorithm is $O(n)$, where n is the number of vertices on the mesh which are evaluated at each step.
The vertex removal operator is topology preserving. This characteristic is desired in certain applications, for example finite elements analysis. But the topology preservation is not mandatory for rendering. Some small holes can be removed from the mesh without problems. But, if there are isolated vertices, these will not be eliminated from the mesh.
An algorithm designed to reduce the mesh topology is proposed in [Schroeder, 1997]. In this algorithms vertex removal is used together with a vertex split, operator which permits to reduce the mesh topology.
The difference between methods which use vertex removal as a simplification operator is the error metric used for measuring the geometric deviation introduced by the vertex removal.

Edge Collapse

Edge collapse (Figure 3.8) is among the most common simplification operations, first proposed in [Hoppe *et al.*, 1993]. This operator replaces an edge by a point. We denote the collapsed edge by: $e = (v_1, v_2) \rightarrow v$.
All the triangles connected to v_1 and v_2 will be now connected to the new vertex, $\bar{v}$. The

triangles which shared the deleted edge e are deleted from the mesh configuration.
Thus, the vertex removal operator simplifies the mesh with one vertex, two faces and three edges if it is an interior edge and with one triangle, two edges and one vertex if it is a boundary edge [Hoppe, 1996], [Wu and Kobbelt, 2003b], [Heckbert and Garland, 1999], [Garland, 1999], [Division and Guéziec, 1996], [Ronfard *et al.*, 1996].
With each iteration, for each candidate edge, the geometric deviation created by its collapse is computed. The edge with the minimum geometric deviation is chosen to be collapsed. The remained edges are then re-evaluated.
An important aspect of the edge collapse operation is the location of the new vertex. The position of the new vertex can influence the quality of the simplified mesh and this influence is more noticeable for drastic simplifications.

In [Hoppe, 1996], the concept of progressive meshes is introduced. This method is an improvement over [Hoppe *et al.*, 1993] in terms of speed and quality of output. The idea is representing the original mesh with a simplified version of it together with a stream of additional information. In this way, different LODs of the original mesh could be reconstructed.
An original mesh $\mathcal{M}$ is represented as:

$$\mathcal{M} = \mathcal{M}_0 + \{(v_{si}, v_{ti}, v_{li}, v_{ri}, A_i)\}, for\ i = 1 \dots N$$

where $\mathcal{M}_0$ is the most simplified version of the original mesh, and $\{(v_{si}, v_{ti}, v_{li}, v_{ri})\}$ is the stream of information which permits the reconstruction of the original mesh or different LODs of it. The simplified mesh is obtained from the original by applying succesive edge collapses.
For the reconstruction of the original mesh from the simplified one, vertex split, the inverse of the edge collapse is used.
For each interior edge collapse, the following information regarding the collapse is recorded: (v_s, v_t, v_l, v_s, A) where v_s is the vertex after edge collapse, v_t is the vertex which, together with v_s formed the collapsed edge. v_l and v_r are vertices which, together with v_t and v_r formed the triangles adjacent to the collapsed edge.
In A is stored the vertex attribute information.
The progressive mesh simplification approach is proposed in [Hoppe, 1997], [Hoppe and Wa, 1998] as well.

The applications for progressive representations are progressive transmission (when data is transmitted over the internet, at reception the quality of the mesh is gradually improved as data is incrementally received), selective refinement (selected regions are represented in more detailed), LOD approximation, mesh compression (memory storage is reduced).

[Schroeder, 1997] proposes another progressive representation scheme based on edge collapse-

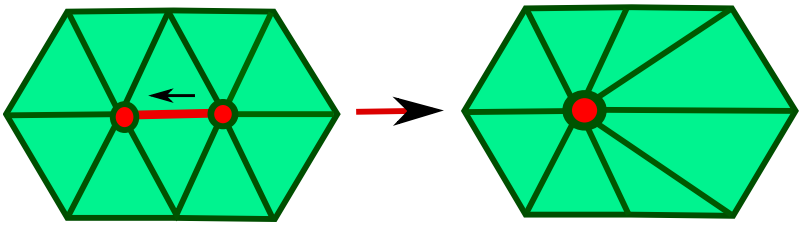


Figure 3.9: *Half-Edge Collapse. The edge is collapsed to one of its endpoints.* The number of vertices is reduced by one, the number of faces is reduced by two and the number of edges is reduced by three.

split and vertices collapse-split. Because the mesh simplification vertex pairs are collapsed, the algorithm permits simplifying the topology. Another interesting property of vertex splitting is that no base mesh is required. To start mesh reconstruction only a few vertices are needed to initialize edge split operations.
Each vertex in the mesh has an error associated. The error metric is based on [Schroeder *et al.*, 1992]. When an edge is collapsed to one of its endpoints, the error of the deleted vertex is transferred to the remaining vertex. For example, $e = (v_1, v_2) \rightarrow v_1$ and the errors associated are e_1 and e_2 the vertex after collapse has the error $e_1 = e_1 + e_2$. Differing from [Schroeder *et al.*, 1992], here the vertex removal is realized by edge collapse. Moreover, re-triangulation is not necessary, an edge collapse operator being used for simplification. Using edge, vertex collapse-split, the algorithm is able to produce progressive mesh representations.
Each algorithm which uses edge collapse as topological operator can be successfully applied for progressive meshes.

Half-Edge Collapse

There are simplified operators which shrink an edge to one of its vertices (Figure3.9) [Hoppe *et al.*, 1993], [Ronfard *et al.*, 1996], [Kobbelt *et al.*, 1998]. These operators are called *half-edge collapse* operators. In this category, both collapses $e = (v_1, v_2) \rightarrow v_1$ and $e = (v_1, v_2) \rightarrow v_2$ are considered possible candidates [Mäntylä, 1987].
The advantage of using a half-edge collapsing operator is that the vertices in the reduced mesh are a subset of the vertices from the original mesh. This leads to a more efficient application of the algorithm for progressive transmission.
The disadvantage is there is no freedom of choosing the new geometry. Thus, the reduced mesh does not fit well with the original one.
Both edge collapse and half-edge collapse operators preserve the topology of the mesh.

Vertices Contraction

Differing from the edge contraction, vertex contraction permits collapsing two vertices which are not connected through an edge but are closer to each other than a user defined threshold distance (Figure 3.10). This operator is useful in applications where topology

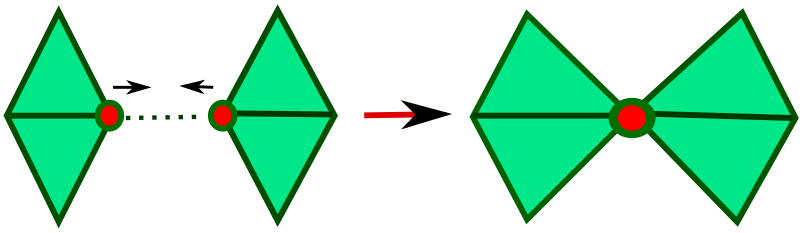


Figure 3.10: *Vertex Pair Contraction. A pair of unconnected vertices are collapsed. The number of vertices is reduced by one, the number of edges and faces remaining constant.*

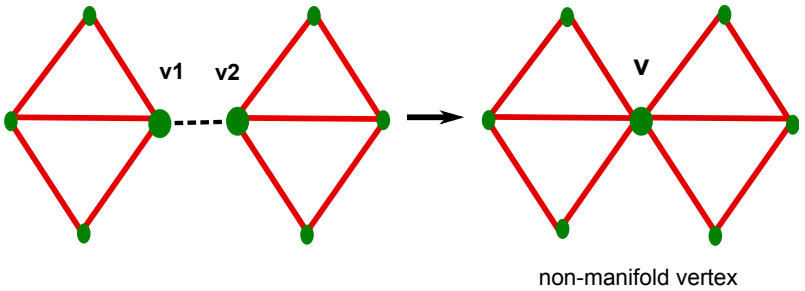


Figure 3.11: *Vertex Pair Contraction. If two vertices such as v_1 and v_2 are collapsed, their collapsed can generate non-manifold vertices, such as veretx v.*

simplification is desired.
For example, for simplifying a mesh with a large number of holes, called a mesh with *sponge topology* [Botsch *et al.*, 2008], applying only the edge contraction operator is not enough and the simplification cannot be efficiently realized. Simplifying this type of mesh with edge and vertices contraction does a better job. Vertex contraction reduces the number of vertices by one, while the number of faces and edges is kept constant. There are certain applications in which real time rendering is more important than preserving the topology and in these situations, vertices contraction can be used.
One of the best algorithms in literature in terms of the balance between quality of the results and speed and which uses vertex pair contraction is proposed in [Garland and Heckbert, 1997].
A pair of vertices includes both vertices which are connected by an edge and all vertex pairs which are not connected by an edge but are closer to each other than a user-defined distance threshold.
A disadvantage of vertex pair contraction is that it can produce non-manifold meshes.

Generalized Pair Contraction

In [Borodin *et al.*, 2003], the concept of vertex pair contraction is extended to more-generalized ones: vertex-edge contraction, vertex-triangle contraction or even edge-edge contraction.
These generalized operators do not permit geometry simplifications, but the topology simplifications. Collapsing a vertex and an edge or a triangle closes holes or connects

neighbouring parts of the mesh which cannot be closed using only vertex-vertex collapsing. There are situations when a vertex from a part of the mesh is close to another part, but far enough to a vertex on that part so that the vertex collapse is possible without causing distortions.
For characterizing the geometric deviation generated by a pair contraction, an error metric is employed. This error metric is the sum of two terms: a quadratic error metric which describes the local deviation between the meshes before and after simplification and a Hausdorff distance which describes the deviation between the original mesh and the actual mesh (the mesh before pair contraction). In this error metric, quadric errors are not accumulated. A candidate pair chosen to be collapsed will be collapsed only if the one-sided Hausdorff distance does not exceed a pre-defined threshold distance. We denote here by *one-sided Hausdorff distance* the maximum distance between the two meshes involved in the computation of the Hausdorff distance.
As candidate pairs are sorted in a priority queue and the one which produces the least geometric deviation is chosen. If the chosen pair fails certain tests, the pair is discarded and inserted in a second priority queue.
Even if this algorithm can produce high-quality results better than the quadratic error metric [Garland and Zhou, 2005], there is no estimation of running time.
Another algorithm based on *vertices contraction* that adopts the *progressive representation* is proposed in [Schroeder, 1997]

Faces Merging

With the algorithms which use faces merging as the simplification operator, coplanar or nearly coplanar faces are grouped in patches. The boundary edges of each patch are then re-triangulated. Actually, each patch is replaced by a simpler version of it. An important aspect how the patches are formed. In [Hinker and Hansen, 1993], faces with parallel or nearly parallel normal vectors are grouped in the same patch. This method is inefficient in terms of reducing complexity for curved surfaces.
In [Kalvin and Taylor, 1996] and [Division *et al.*, 1993], all faces which are closer to their approximated plane than an error tolerance are grouped into patches, called in the papers *superfaces*.
Once patches are created, their boundary edges are simplified using Douglas-Peucker algorithm [Ramer *et al.*, 1972] and re-triangulated.
Different from the method of [Kalvin and Taylor, 1996], which creates patches based on the normals vectors, this method is less sensitive to noise, patches being constructed based on the distance from face to plane.

Inspired by the *vertex removal* operator, the *triangle removal* operator is proposed in [Hamann, 1994]. A weight given by the curvature of the triangle vertices is associated to each triangle and for each iteration, the triangle with the lowest weight is replaced by a vertex.

All triangles which share an edge with the replaced triangle are deleted. The *Triangle removal* operator reduces the complexity of a mesh with two vertices, four faces and six edges. This operator is unable to reduce the topology and works better on planar or nearly planar surfaces.

The triangle removal operator is used in [Varshney *et al.*, 1995], [Varshney, 1994] as well, but by this approach the mesh to be simplified is bounded by two envelopes which guarantee the simplified mesh will be deviated by no more than a user-defined tolerance.
Vertex removal is the simplification operator used in [Hamann and Chen, 1994].

3.2.2 Direct Simplification

Direct simplification algorithms reduce the complexity of the mesh in one single step. The simplification is achieved by grouping the mesh's vertices into clusters. A representative vertex for each cluster is chosen and using the representative vertices, a retriangulation is realised.
One of the advantages of the vertex clustering approach is speed. Only the representative vertex computation is time consuming. Furthermore, these algorithms are robust and both manifold and non-manifold inputs can be used. The level of simplification can be controlled by the size of the clusters.
There are certain inconveniences. One of them is that the quality of the output is not satisfactory, and more so at low levels of details.

Direct simplification algorithms are unable to create LOD's representations.
Another drawback of vertex clustering is that the topology is not preserved. If two geodesic disconnected parts of a surface fall inside the same cell, they will converge to a single point, and the topology is altered. This can be considered an advantage for simplifying meshes with sponge-like topology where simplification by using an algorithm with topology preservation is almost impossible.
In vertex clustering simplification, the global error introduced by approximation is contrroled by the size of the clusters.

Vertex Clustering

A classification of vertex clustering algorithms is based on how clusters are created and how the representative vertex for each cluster is computed.
To cluster the vertices, a bounding box is placed around the mesh and uniformly subdivided into three dimensional cells. Each cell in the grid must have a diameter less than a predefined tolerance,ξ. In this manner, the vertices of the mesh are grouped into cells. All vertices inside a cell form a cluster.
A representative vertex for each cluster is chosen, and all the vertices inside a cluster are assigned to the representative vertex. A cluster of vertices is reduced to a single vertex. All

triangles that fall in the same cell are degenerated into a point and all triangles of which two of their vertices are in the same cell are degenerated into an edge. In this manner, complexity is reduced.

The method was originally proposed in [Rossignac and Borrell, 1992]. By this approach, a bounding box is overlaid on the mesh, and uniformly subdivided into 3D cells. A weight is assigned to each vertex. If a vertex belongs to a large faces or to a region of high curvature, the respective vertex has a bigger weight. And a small weight is attached to small faces or vertices from almost flat regions.
After vertices are classified according to their weights, a representative vertex for each cell is chosen. The representative vertex is the vertex with the maximum weight from the respective cell. All the vertices which belong to a 3D cell will be collapsed to the representative vertex.
Triangles which have all vertices inside the same cell are degenerated and eliminated from the mesh. In this manner, the complexity of the mesh is reduced. Because the clusters are uniform, there are situations when feature vertices fall inside the same cluster and consequently are reduced to a single vertex. In this situation the features are not preserved and this represents a drawback of uniform clustering.

A non-uniform clustering is proposed in [Low and Tan, 1997]. This approach is called *floating-cell clustering*. The advantage of *floating-cell clustering* is that the positions of the cells are controlled by the weights of the vertices. Moreover, the size of the cells is controlled by the user.
All vertices are ordered by their weights. The vertex with the highest weight becomes the center of a 3D cell. This cell has a user-defined size. All vertices which lie inside the same cell are collapsed to the representative vertex and degenerated triangles are eliminated. If a vertex is at the intersection of multiple cells, it is attributed to the cell with the closest center.
The same steps are applied for the remaining vertices.
The weight vertices computation is improved in this algorithm, compared to [Rossignac and Borrell, 1992], but running time is increased because of the new weight computations. Because the clusters are non uniform, this method preserves the sharp features and appearance is improved. However, complexity is higher than in [Rossignac and Borrell, 1992].

[Valette and Chassery, 2004] proposes a simplification algorithm called *Approximated Centroidal Voronoi Diagram* (ACVD) based on vertex clustering which involves the Centroidal Voronoi Diagram to build the clusters. In this context, a *Centroidal Voronoi region* contains a subset of connected triangles from the original mesh. Two adjacent clusters are delimitated by a subset of edges, called *boundary edges*.
During the first step, the number of clusters is fixed and one face of the mesh is randomly allocated to each cluster. Looping on all the *boundary edges*, all faces which share

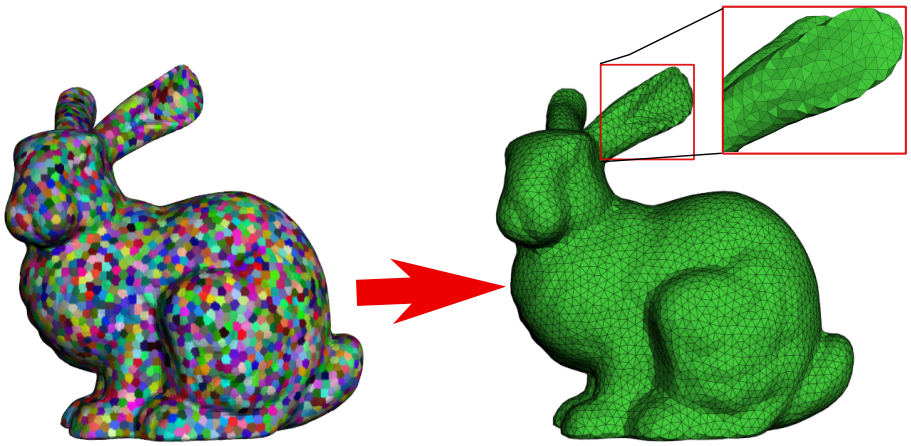


Figure 3.12: *Mesh simplification using ACVD method.* In the right part is represented the original Bunny (36 000 vertices) with 5000 clusters. In the left part, its simplified version with 5000 vertices. A representative vertex for each cluster was created. The triangulation is realized using Delaunay triangulation.

a boundary edge and are not allocated to a cluster will be allocated to the cluster the boundary edge belongs to. In this manner, all faces of the mesh are initially allocated to clusters. The energy term which defines the distribution of the mesh's triangles into clusters is defined as:

$$E = \sum_{i=0}^{n-1} \left(\sum_{t \in C_i} a_t c_t^2 - \frac{\| \sum_{t \in C_i} a_t c_t \|^2}{\sum_{t \in C_i} a_t} \right) \tag{3.24}$$

where n is the number of clusters, t is a triangle in the cluster C_i, a_t is the area of the triangle and c_t is the centroid of the triangle.

To create clusters which simulate the *Centroidal Voronoi Regions*, a minimization energy algorithm is iteratively applied. For each boundary edge, the energy function is computed in three situations: when the triangles which share the boundary edge belong to two different clusters, when they belong to the first cluster and when they belong to the second one.

The configuration which produces the minimum energy function is chosen. For example, if there is a boundary edge e, which bounds two clusters, C_m and C_n, and shares the triangles $t_m \in C_m$ and $t_n \in C_n$, the energy function E is computed for the initial configuration, for the case when both $t_m, t_n \in C_m$ and for the case when both $t_m, t_n \in C_n$. When the final configuration of the clusters is chosen, a triangulation is applied using Delaunay triangulation. The vertices of the output are the vertices from the original mesh which are closest to the clusters' centroids.

Being an algorithm based on vertex clustering, the algorithm simplifies the geometry and the topology of t mesh as well. Because in the energy term each triangle is weighted by its area, the clusters are uniform.

Because the clusters are uniform, sharp features and flat regions are reduced as well and

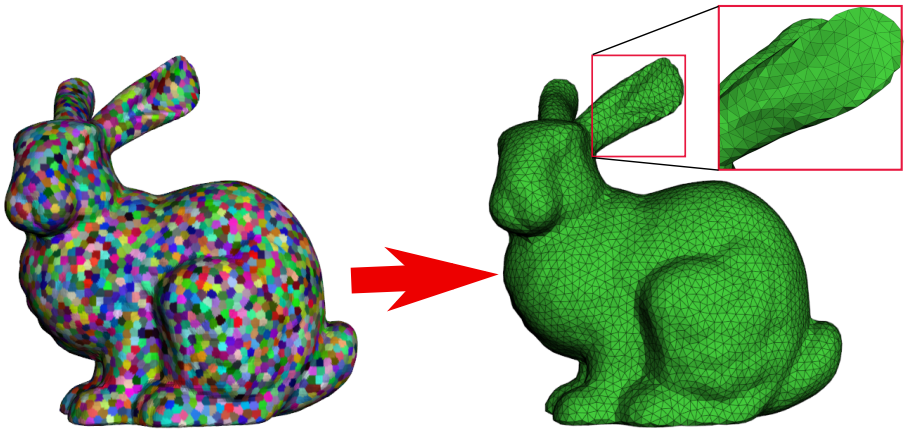


Figure 3.13: *Mesh simplification using ACVDQ method.* On the right part the original Bunny (36 000 vertices) is represented with 5000 clusters. On the left part, its simplified version with 5000 vertices. For each cluster a representative vertex was found. The triangulation is realized with Delaunay triangulation.

the algorithm produces uniform output tessellations (see Figure 3.12).

To address this drawback, the authors of ACVD proposed a simplification algorithm based on variable size clusters which takes into consideration the mesh curvature information [Valette *et al.*, 2005]. The size of the clusters varies with the density of the vertices from the mesh. Moreover, the density of the vertices can be an indicator for the curvature of the mesh: a region with high curvature has more vertices than a flat one.
Comparing the results from Figure 3.12 and Figure 3.13, features such as the ears are better preserved in the simplified Bunny from Figure 3.13.

Because it is a very fast simplification algorithm, it can be successfully applied in video games, where interactivity is more important than the accuracy of the representation.

Out-of-Core Simplification

Out-of-core simplification algorithms [Lindstrom, 2000], [Lindstrom and Silva, 2001], [Wu and Kobbelt, 2003b] are designed to simplify meshes too big to fit entirely in the main memory.
There are two methods for handling this type of simplification:

- to subdivide the mesh into unconnected patches and each patch is simplified independently in-core. All simplified patches are joined together to obtain the simplified mesh.

- to design an algorithm which uses only limited connectivity information for simplification.

The first approach is used in [Bernardini *et al.*, 2002]. The mesh is subdivided into unconnected patches, each patch is processed in-core, but the boundaries are kept unchanged. When all patches have been simplified, they are joined together and a new simplification starts in order to simplify the boundaries that remained unsimplified before. Splitting and joining together the patches after simplification is fairly expensive and makes the method less attractive. Moreso, the artifacts are introduced along the boundaries of the patches. Mesh cutting is the out-of-core simplification method used in [Hoppe, 1998], [Prince, 2000] and in [Ho *et al.*, 2001] for compression of large meshes.

One of the earliest out-of-core methods which uses limited connectivity for simplifying a mesh is proposed in [Lindstrom, 2000].
For each face, the quadratic matrix is computed as in [Garland and Heckbert, 1997]:

$$Q = \begin{pmatrix} A & -b \\ -A^T & c \end{pmatrix}$$

where A is the same from Egn. 3.7.
The author addresses the situations where A is not invertible (when the surface is too flat).
To avoid these situations, the *singular value decomposition* of A is used.
Thus,

$$A = U\Sigma V^T \tag{3.25}$$

with

$$\Sigma^+_{ij} = \begin{cases} \frac{1}{\sigma_i} \text{ if } 1 \leq i \leq r \\ 0 \text{ if } r < i \leq n \end{cases} \tag{3.26}$$

and

$$\Sigma \cdot \Sigma^+ = I \tag{3.27}$$

σ_i are the eigenvalues of $A^T A$.
If Σ is invertible, $\Sigma^+ = \Sigma^{-1}$.
The author uses this error metric for determining the position of the representative vertex for a cluster (see Section 3.2.2).
The representative vertex is the vertex closest to the center of the cluster:

$$x = \hat{x} + U\Sigma^+ U^T(b - A\hat{x}) \tag{3.28}$$

where $\hat{x}$ is the center of the cluster.
In order to have random access to different parts of the mesh, each triangle is represented in the out-of-core memory as a list of its vertices' coordinates. After the cluster grid is constructed each triangle is fetched from the out-of-core memory and its quadric is computed. A dynamic hash table maps clusters to the vertices, thus allowing a fast access.

The quadric for each cluster is the sum of the quadrics of vertices which belong to that cluster.
In this manner, the algorithm computes the clusters' representative vertices using limited connectivity information and polygons soup. However disadvantage of limited connectivity is a lower quality of output.

[Lindstrom and Silva, 2001] proposes an algorithm which does not ask for the all simplified mesh to be in the main memory but uses an external data structure where all necessary information for computing the quadric is saved. After the clusters are created, the information is reordered using sorting algorithm. Then, by using a single loop over the sorted file, the clusters quadrics and representative vertices are computed. This way, the amount of memory remains constant and is independent from the size of the input or output.

An out-of-core approach is proposed in [Isenburg *et al.*, 2003]. The simplification is based on *mesh processing sequences*. Using mesh processing sequences, a mesh is presented as an interleaved sequence of indexed triangles and vertices. Each edge and vertex has a status about its visit through the main memory.
The processing sequences approach from [Lindstrom and Silva, 2001] reduces the memory necessary for clustering vertices. Moreover, the quality of the output is improved. In this approach, the amount of active work is reduced compared to [Wu and Kobbelt, 2003b].

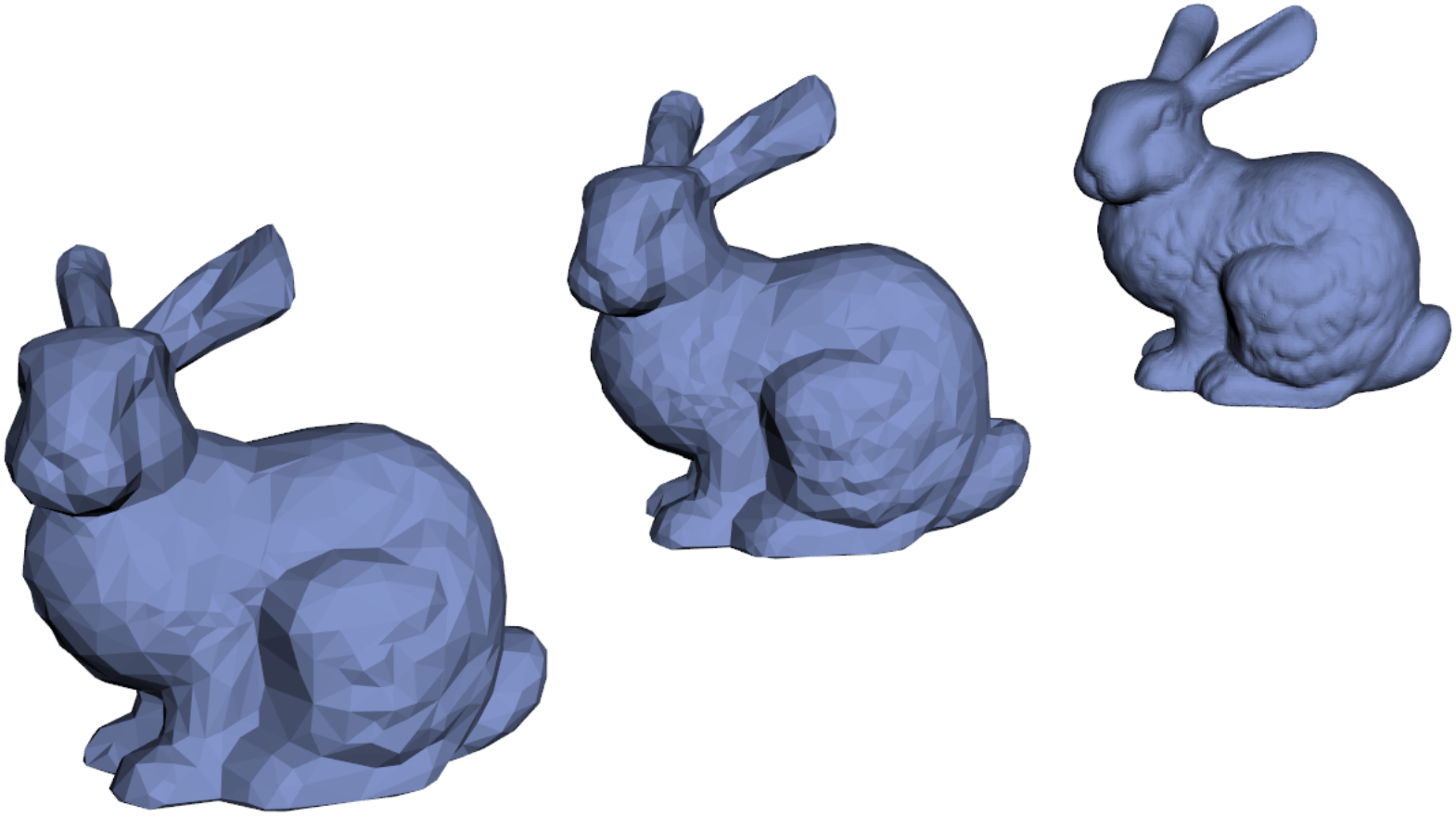
In the out-of-core approach proposed in [Shaffer and Garland, 2001], the mesh is densely partitioned into uniformly grid clusters in out-of-core memory. This intermediate mesh is used as an input for the edge-collapse simplification (see Section 3.2.1), operation which is realized in-core.
For both cluster simplification and edge collapse, the quadratic error metric is used.

An extension of [Lindstrom and Silva, 2001] implemented on GPU (graphics processing unit) is proposed in [DeCoro and Tatarchuk, 2007]. The vertex clustering method (see Section 3.2.2) is adapted to the GPU by using a *vertex shader* and a *geometry shader* [Blythe, 2006]. The vertex shader assigns each vertex to a cluster and computes the vertex representative for each cluster. The geometric shader computes the quadric for each triangle and assigns that value to each vertex processed by the vertex shader. The algorithm allows varying sampling rates for the simplification by deforming the cluster grid during simplification. The deformation of the cluster grid is achieved by using a non-rigid warping function. This way, the method permits view-dependent simplification necessary for applications where more detail is necessary in regions closer to the viewer. In addition, region-of-interest simplification is possible with non-uniform clustering. Selected regions rendered with greater detail than the others.
Simplifications done with the GPU approach are up to 15 times faster than with CPU

approach, for identical models. Moreover, memory requirements are constant and lower. The algorithm produces level-of-detail and out-of-core representation as well.

Chapter 4

Contributions



The goal of simplification algorithms is to reduce the complexity while keeping a high fidelity to the original mesh.
We proposed designing a simplification algorithm the main purpose of which is accuracy. We noticed that simplification algorithms existing in literature have a lack of accuracy in measuring the geometric deviation introduced by a simplification operator.
For example, the error is computed as the sum of the squared distances from a vertex to the planes of the triangles inherited by that vertex as in [Garland and Heckbert, 1997]. For almost planar surfaces, this error metric can work well, but not for curved surfaces. The distance from a vertex to the supporting plane of a triangle can be smaller than the distance to the triangle. In this situation, the actual distance is underestimated. On the other hand, the error metric is computed as the distance between a removed vertex and the approximated plane of its neighboring vertices, as in [Schroeder, 1997]. This error metric can give accurate results for planar or almost planar surfaces but faulty estimations of the geometric error for curved surfaces. The more neighboring vertices around the removed vertex are situated in different planes (the situation of curved surfaces), the more the error metric is underestimated.
Moreso, in this method, the geometric error is computed by reffering to the previous mesh geometry and not to the original one. Thus, the estimated geometric deviation is only an approximation.
There are algorithms which use as geometric error metric the maximum distances from the simplified surface to the original one [Ciampalini *et al.*, 1997]. In these algorithms, the error is accumulated after each simplification. This way, the error is reffered to the original model. The maximum distance between two meshes is not sufficient for evaluating the geometric deviation between them. For example, for two surfaces as in Figure 4.1, if the geometric deviation is the maximum distance between M_o and M_s, the error is d_{max}, which is not the real distance between the two surfaces. Using this measure as a criterion for mesh simplification can lead to rough simplifications as in Figure 4.2.
The Hausdorff distance used for simplification as an error metric leads to rough approximations [Klein *et al.*, 1996]. On the other hand, this is one of the few methods which takes into account the symmetric distance between two surfaces.
Measuring the distance between two meshes in a symmetric fashion, from the simplified surface to the original one and from the original to the simplified is useful for boundary preservation and preserving the creases of a mesh or some isolated islands.

Starting with the drawbacks of the methods in literature, we proposed designing a simplification algorithm which works on both manifold and non-manifold meshes and which is able to simplify the geometry of the mesh in an accurate and symmetric way. And last but not least, to compute the geometric deviation from the original mesh.
For achieving accuracy, the error metric we use is the weighted sum of squared distances from the points of the source surface to the destination surface faces and not to the supporting or approximated planes of these.

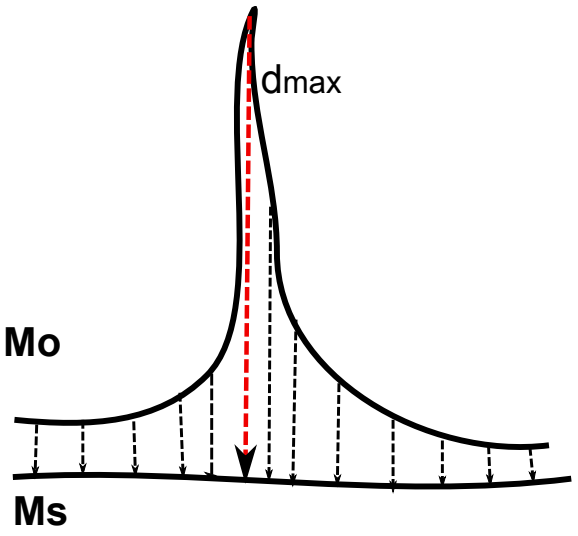


Figure 4.1: *The representation of the maximum distance between two surfaces.*

In addition, preserving bounadries and features as creases, we measure the distance between the surfaces (simplified and original) in a symmetric fashion.
The simplification operator we choose is the edge collapse. This operator works well on both manifold and non-manifold meshes and preserves mesh topology.
Topology preservation is desirable for applications such as medical visualization or mechanical simulations. But topology reduction is desirable for eliminating the artefacts presented in a mesh.
We proposed designing a simplifiction method which by disregarding running time is more accurate. Our simplification algorithm is designed to make the simplifications off-line, because we are interested in the quality of the simplification, so, there is no time constraint. The condition for stopping our simplification algorithm is determined by the complexity or the output model and not by the error introduced during simplification.

Next, we will present the outline of our simplification algorithm, with a detailed description of the error metrics chosen for describing the deviation introduced by edge collapses.

4.1 Proposed Simplification Algorithm: overview

The simplification algorithm we propose is based on an iterative edge collapse operation.

For computing the geometric deviation introduced by a possible edge collapse, we use two measures, an *Accurate Measure of Quadratic Error* (AMQE) and a *Symmetric Measure of Quadratic Error*(SMQE).
At each step of the simplification, all possible collapses are simulated and introduced in a priority queue. The edge which produces by its collapse the minimum geometric deviation is chosen to be collapsed. We use the term simulation because at each step of the simpli-

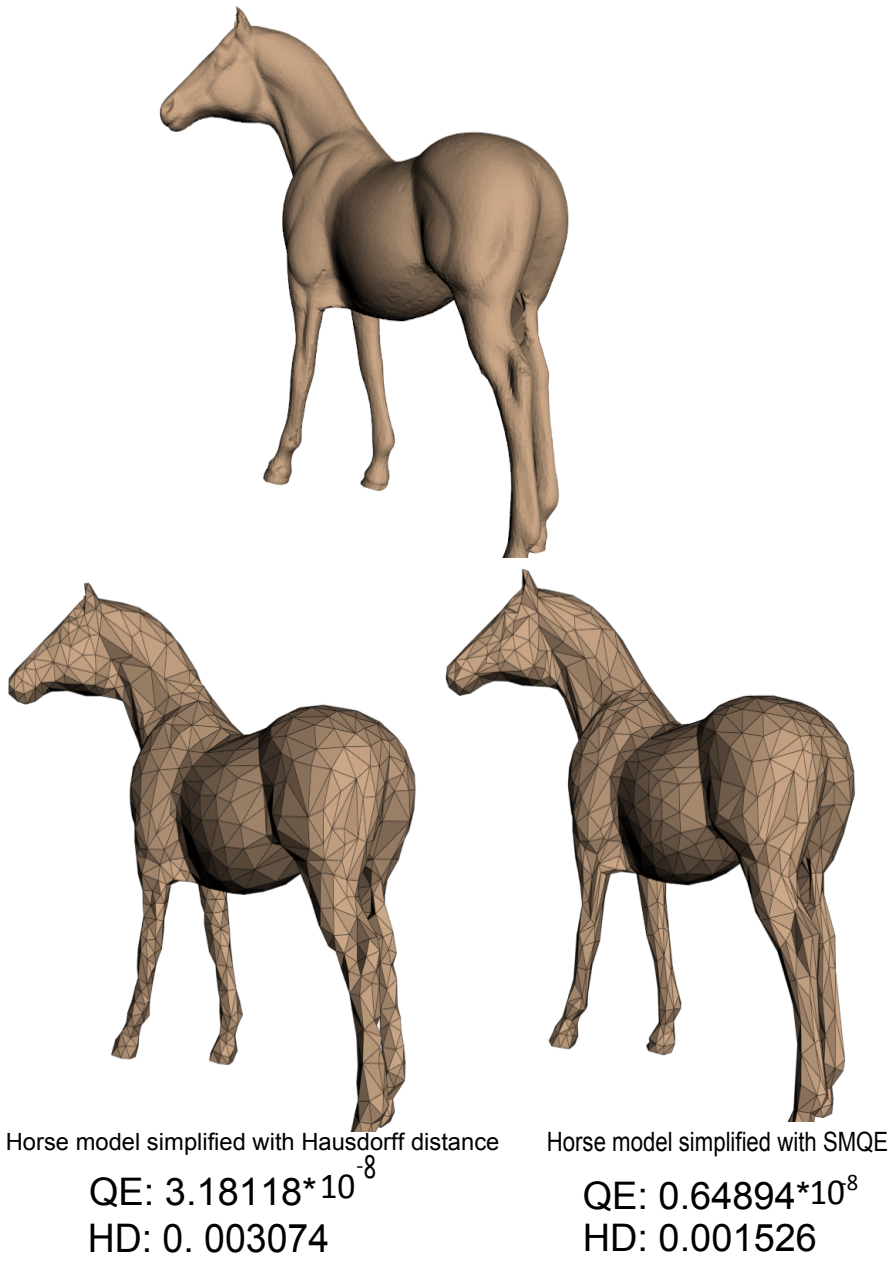


Figure 4.2: *The simplification of Horse model using the Hausdorff distance (right) and SMQE (left) as error metric.*

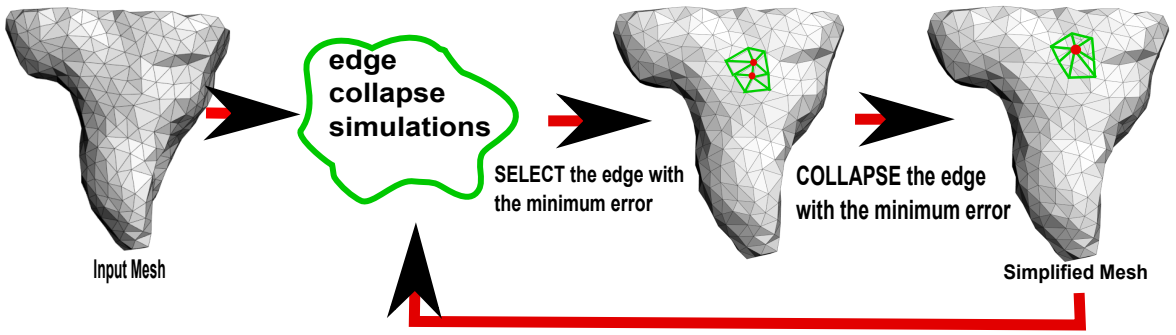


Figure 4.3: *Overview of the simplification algorithm.*

fication we collapse the edges only virutally in order to obtain the geometric error they would introduce by their collapse and afterwords, we collapse the edge which produces the minimum deviation.

The simplification algorithm is outlined as follows:

1. On the original mesh, all possible edge collapses are simulated.
2. The edge which introduces by its collapse the minimum geometric deviation is chosen to be collapsed.
3. The edge is collapsed to a single vertex $e = (v_1, v_2) \rightarrow \bar{v}$. The following operations are performed:
   - the position of the resulting vertex, $\bar{v}$ is computed: $(v_1, v_2 \rightarrow \bar{v})$ by minimizing the quadratic error metric between the vertex and the original mesh.
   - all the faces which were connected to v_1 and v_2 before are now connected to $\bar{v}$.
   - the resulting vertex, $\bar{v}$ takes the id (identification index) v_1.
   - the vertex v_2 and the faces which shared the collapsed edge are eliminated.
   - the errors are reevaluated for all edges of the simplified model.
4. All these steps are repeated until the stop condition is reached.

We choose edge collapse as the operation for reducing the mesh complexity for the following reasons: an edge collapse performs local modifications to the mesh that permits a good control of the error introduced by simplifications; it produces level of detail representation; it reduces the mesh geometry while preserving its topology; it works both on manifold and non-manifold meshes.

The error metric used for measuring the geometric deviation introduced by an edge collpase has an important role to play in the quality of the simplified mesh while the goal of simplification is reducing the complexity while keeping the simplified mesh as close as

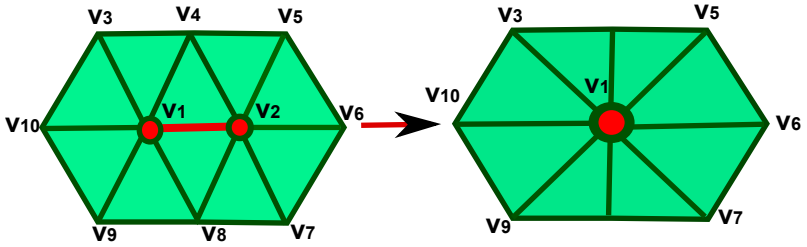


Figure 4.4: *Interior Edge Collapse. One edge is collapse to a single vertex. Triangles which share the deleted edge are deleted. The number of vertices is reduced by one, the number of faces by two and the number of edges by three.*

possible to the original.
In this thesis, the fidelity of the simplified mesh to the original is expressed in terms of the geometric deviation between two meshes.
Next, we will present the geometric errors used for describing the geometric deviation introduced by an edge collapse.

4.2 Edge Collapse Simulations

The edge collapse simulation is probably the most important part of the simplification algorithm.
During this phase, the algorithm simulates all possible edge collapses and computes the errors which might be introduced by the possible edge collapses.
All non-manifold edges are considered as candidate edges. After the simulation of all possible collapses, the edge which produces the least geometric deviation is choosen to be collapsed.
An edge collapse changes both the geometry and the compexity of the mesh. The complexity of the mesh is reduced by one vertex, two faces and three edges each time when an interior edge is collapsed (Figure 4.4). When a boundary edge is collapsed, the complexity is reduced by one vertex, one face and two edges.
The geometry of the mesh is changed because the edge is collapsed to a new vertex, $\bar{v}$.

Our simplification algorithm is quite similar to the simplification algorithm proposed in [Garland and Heckbert, 1997], the only difference is the quadratic error metric involved to evaluate the geometric deviation introduced by an edge collapse.
Next, we will present our first proposed error metric involved to evaluate the geometric deviation introduced by an edge collapse.

4.3 First Error Metric used: Accurately Measured Quadratic Error

The first error metric we proposed in order to evaluate the geometric deviation introduced by an edge collapse is based on the distances in one direction, from the simplified mesh to the original one. We called this error metric: **A**ccurate **M**easure of **Q**uadratic **E**rror (**AMQE**).
The accuracy of this error metric is provided by two features: the global measurement of the deviation between the meshes (the simplified mesh and the original one) and the use of the original mesh as reference.

Next, we denote by $\mathcal{M}_o$ the original mesh and by $\mathcal{M}_s$ the simplified one.
We denoted by $preKernel(\mathbf{e})$ the set of triangles which share the endpoints of the edge $\mathbf{e}$ (the red patch from Figure 4.5, left).
The $postKernel(\mathbf{e})$ represents the set of triangles which share the vertex resulting after the edge $\mathbf{e}$ collapsed.
AMQE is defined as the triangle area weighted sum of the squared distances from the simplified mesh to the original, as follows:

$$\mathbf{AMQE}(\mathcal{M}_s, \mathcal{M}_o) = \frac{\sum_{i=1}^{F} d_{\Delta_i}}{\sum_{i=1}^{F} a_{\Delta_i}} = \frac{D(\mathcal{M}_s, \mathcal{M}_o)}{A(\mathcal{M}_s)} \tag{4.1}$$

where:

- F is the number of faces in $\mathcal{M}_s$
- d_{Δ_i} is the distance from the face Δ_i to $\mathcal{M}_o$
- $D(\mathcal{M}_s, \mathcal{M}_o)$ is the sum of distances from the mesh $\mathcal{M}_s$ to $\mathcal{M}_o$
- a_{Δ_i} is the area of the face Δ_i
- $A(\mathcal{M}_s)$ is the area of $\mathcal{M}_s$

When an edge collapse is simulated, only the faces of the *preKernel(e)* are virtually modified. The rest of the faces remain unchanged. Thus, only the distances between the modified faces and the mesh have to be recomputed. The distances for the rest of the faces are unchanged.
Thus, the AMQE for an edge collapse is computed as follows:

$$AMQE(\mathcal{M}'_s, \mathcal{M}_o) = \frac{D(\mathcal{M}_s, \mathcal{M}_o) - \sum_{m \in preKernel} d_m + \sum_{n \in postKernel} d_n}{A(\mathcal{M}_s) - \sum_{m \in preKernel} a_m + \sum_{n \in postKernel} a_n} \tag{4.2}$$

where:

- $\mathcal{M}'_s$ is $\mathcal{M}_s$ with one edge simplified.

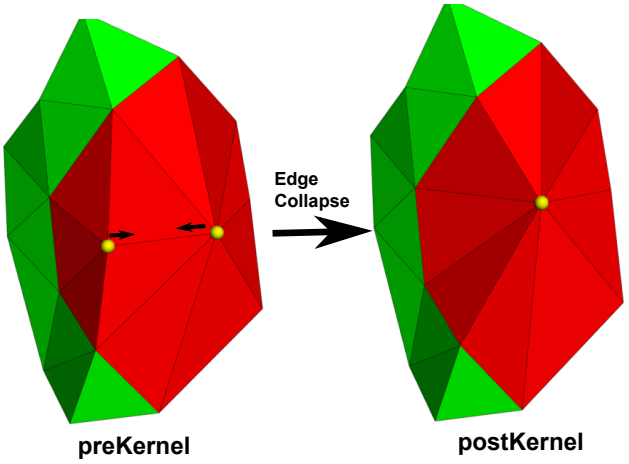


Figure 4.5

- $D(\mathcal{M}_s, \mathcal{M}_o)$ and $A(\mathcal{M}_s)$ taken from Eqn. 4.1.
- $\sum_{m \in preKernel} d_m$ is the sum of the squared distances of the triangles of $preKernel$
- $\sum_{n \in postKernel} d_n$ is the sum of the squared distances of the triangles of $postKernel$.
- $\sum_{i=1}^{F} a_{\Delta_i}$ is the area of the simplified mesh, $\mathcal{M}_s$
- $\sum_{m \in preKernel} a_m$ is the area of $preKernel$.
- $\sum_{n \in postKernel} a_n$ is the area of $postKernel$.

In Eqn. 4.2, the terms $D(\mathcal{M}_s, \mathcal{M}_o)$ and $A(\mathcal{M}_s)$ are already known from the previous step, at each step being computed only the triangles modified by an edge collapse (the triangles in the *preKernel* are modified and replaced by the triangles in the *postKernel*). Thus, for each edge collapse simulation, the computation is reduced to the triangles affected by an edge collapse.

Triangle Distance

d_{Δ_i} from Eqn. 4.1 is the sum of the weighted squared distances from a triangle Δ_i of the simplified mesh, $\mathcal{M}_s$ to the original mesh, $\mathcal{M}_o$.
Approximating the deviation between two surfaces only taking into consideration the distances between the vertices of the simplified mesh to the original mesh, in some situations could lead to underestimationg or overestimating of the real distance.
For example, in Figure 4.6 we want to approximate the deviation between the two curves: M_s and M_o. If we approximate the distances between the curves, M_o and M_s taking into consideration the distances from the vertices of M_s (the red points) to the curve M_o, the distance is overestimated (Figure 4.6, left part) or underestimated (Figure 4.6, right part). Therefore, in order to obtain an accurate estimation of the distance between the two curves, we add sample points on the simplified curve (Figure 4.7).

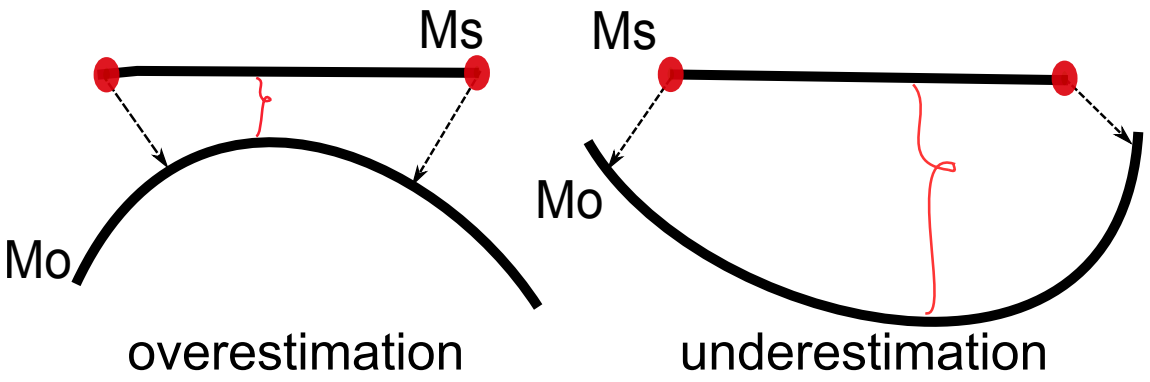


Figure 4.6: *Approximation of the distance between two curves taking into consideration only the vertices (red points) of the simplified curve M_s*

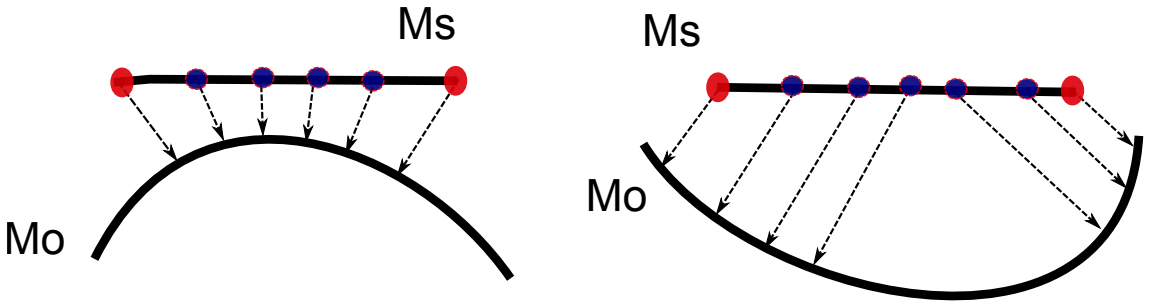


Figure 4.7: *Improvement of the approximated distance between two curves by adding sample points.*

In a 3D situation, we have to add sample points over the triangles of the simplified mesh for which we compute the distance (Figure 4.10).

Triangle Subdivision

For simplicity, we create the sample points of a triangle using 1 : 4 subdivision iterations. At each 1 : 4 subdivision, the triangle is divided into four identical triangles (Figure 4.8).
The sampling of each triangle Δ of $\mathcal{M}_s$ is done by iteratively *1:4* subdivisions of the triangle. We denote the number of subdivisions by *noSub*.

The number of *1:4* subdivisions is selected in order to keep the proportionality between the number of faces of the original mesh and the number of faces of the simplified one.

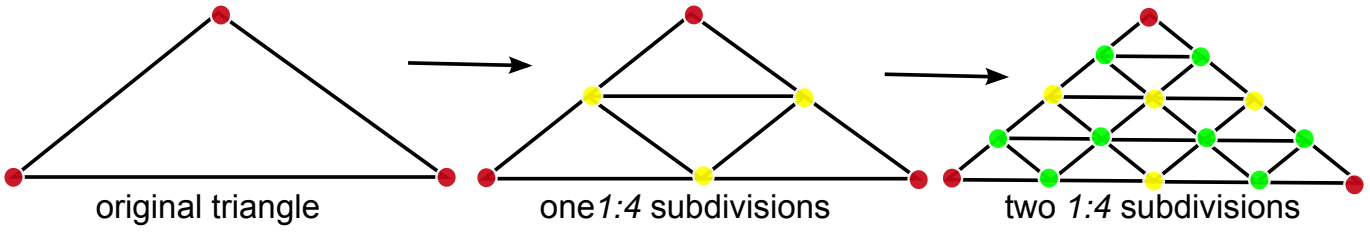


Figure 4.8: *Iterative 1:4 subdivisions of a triangle.*

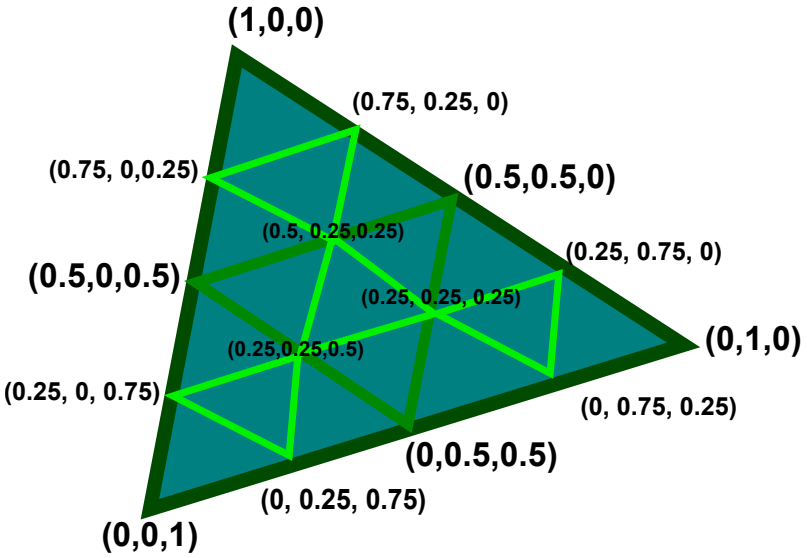


Figure 4.9: *Two 1:4 subdivisions of a triangle and its barycentric coordinates.*

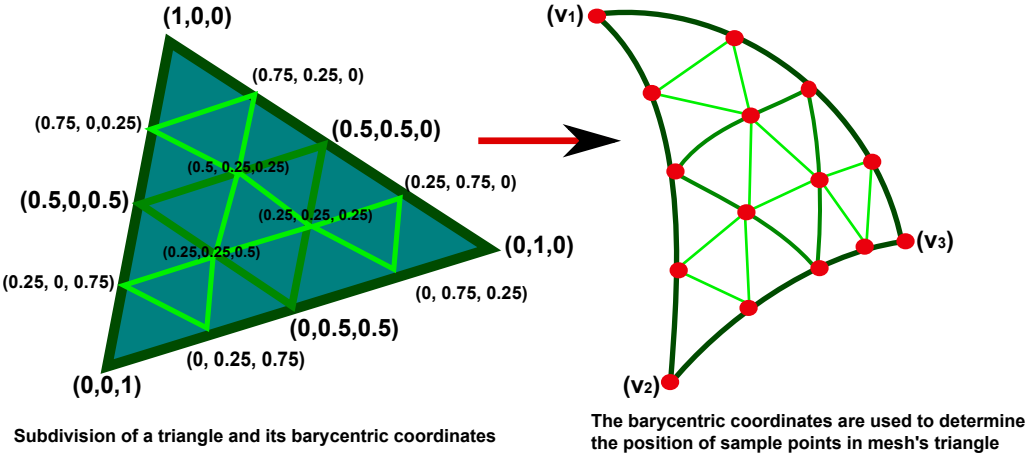


Figure 4.10: *Two 1:4 subdivisions of a triangle and its barycentric coordinates.*

We denoted by $\mathcal{F}_s$ the number of faces on the simplified mesh and by $\mathcal{F}_o$ the number of faces of the original one.
The number of *1:4* subdivisions $noSub$ is obtained as follows:

$$\mathcal{F}_s \cdot 4^{noSub} = \mathcal{F}_o \tag{4.3}$$

$$4^{noSub} = \frac{\mathcal{F}_o}{\mathcal{F}_s} \tag{4.4}$$

$$noSub = floor(\frac{1}{\log 4} \cdot \log \frac{\mathcal{F}_o}{\mathcal{F}_s}) \tag{4.5}$$

According to the number of subdivisions, a triangle is subdivided into 4^{noSub} sub-triangles.

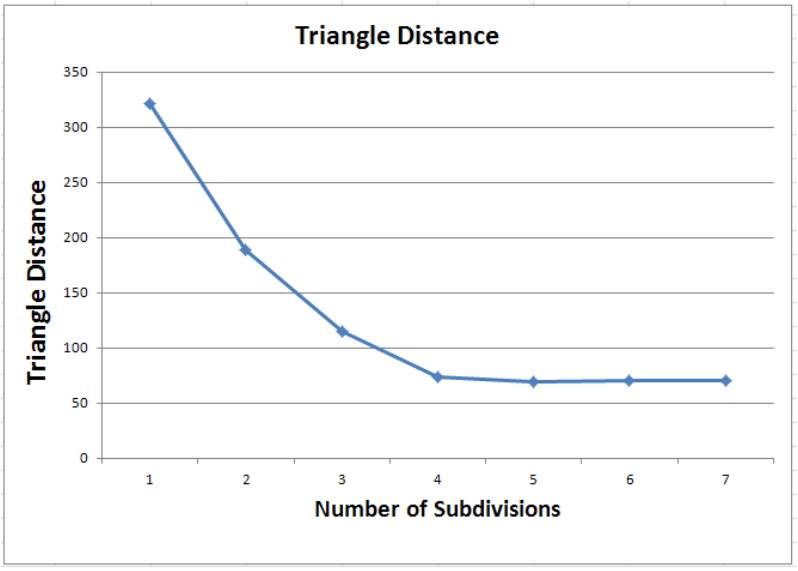


Figure 4.11: *The distance for one triangle accordingly to the number of subdivisions of the triangle.*

The number of sample points for each triangle is (see for example Figure 4.10):

$$n = \frac{(2^{noSub} + 1) \cdot (2^{noSub} + 2)}{2} \tag{4.6}$$

To the vertices of each triangle we associate the following barycentric coordinates: (1,0,0), (0,1,0) and (0,0,1) (see Figure 4.9). The position of the sample points is computed using the barycentric coordinates. Thus, the position of each sample point will be a weigthed sum of the triangle's vertex coordinates, where the weight is given by the barycentric coordinates associated to the point as in Figure 4.10.
We call *sub-triangles* all cells obtained by the subdivision of a triangle.
After subdividing the triangle, we compute the distance from the sample points to the original mesh.
To each sub-triangle we associate a weight and a distance.

In Figure 4.11 it is shown the distance between one triangle of the simplified mesh, $\mathcal{M}_s$ and the original mesh, $\mathcal{M}_o$ and its dependence on the number of subdivisions.
By increasing the number of subdivisions, the distance can begin to decrease (Figure 4.11) or to increase (Figure 4.12), depending on the mesh's geometry, leading to a stable point. In Figure 4.11 the distance begins to stabilize after the 4^{th} subdivisions. In this case, it is useless increasing the number of subdivisions above 4, because the algorithm begins to be time-consuming (Figure 4.13).

We choose to obtain the sample points by *1:4* triangle subdivisions because of its sim-

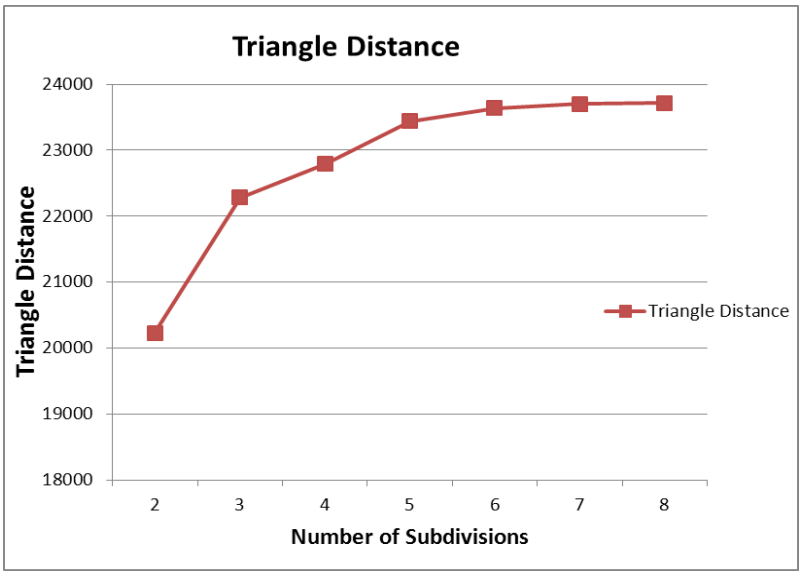


Figure 4.12: *The distance for one triangle accordingly to the number of subdivisions of the triangle.*

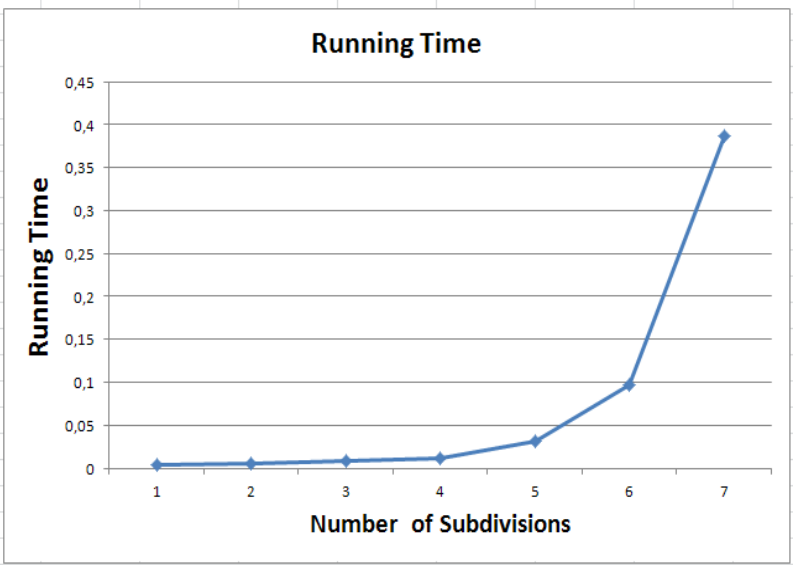


Figure 4.13: *The running time to compute the distance for one triangle accordingly to the number of subdivisions of the triangle.*

plicity. In this manner, the triangle is easily split into sub-triangles with the same areas. This leads to simpler computations.

Thus, the distance of a triangle of the simplified mesh is the weighted sum of the squared distances of its sub-triangles (or the triangles obtained after subdivision).
The weight is the area of the sub-triangles. The distance for a sub-triangle is approximated with the median of the distances from the points of the sub-triangle to the original mesh. More exactly, we compute the distances from all sample points of a triangle to the original mesh and for all three sample points which form a triangle, we approximate their distance to only one value, which is the median of these distances. We choose to use the median from the reasons of simplicity.

Sub-triangle Distance

As we stated above, the coordinates of the points of a sub-triangle are computed using the barycentric coordinates.
For example, if we have the vertices v_1 and v_2 from Figure 4.10, the coordinates of the point which has the barycentric coordinates $(0.5, 0.5, 0)$ are:

$$\begin{aligned} p_{ij_x} &= (v_{1x} + v_{2x}) \cdot 0.5 \\ p_{ij_y} &= (v_{1y} + v_{2y}) \cdot 0.5 \\ p_{ij_z} &= 0 \end{aligned}$$

Thus, the distance for a sub-triangle Δ_i is computed as follows:

$$d(\Delta_i, \mathcal{M}o) = \frac{1}{3} \sum_{j=1}^{3} d(p_{ij}, \mathcal{M}o) \tag{4.7}$$

where:

- p_{ij} is one of the points of the sub-triangle Δ_i
- $d(p_{ij}, \mathcal{M}o)$ is the least distance from the point p_{ij} to the surface $\mathcal{M}_o$

$$d(p_{ij}, \mathcal{M}o) = min_{v \in V}(\|p_{ij} - v\|^2) \tag{4.8}$$

where:

- V is the number of vertices of $\mathcal{M}_o$
- v is a vertex of $\mathcal{M}_o$
- $\| \cdot \|$ is the Euclidean distance between two points

Each sub-triangle is equivalent to a distance which is the median average of the distances of its points.

In order to simplify the computation, we consider the distance for a sub-triangle as being the median average of the distances of its points.

Triangle Area

The area of a subtriangle obtained after the triangle subdivision is computed using Euclidean geometry (Heron's formula).
For a subtriangle Δ_i with the lengths of its sides a, b and c, the area is:

$$area(\Delta_i) = \frac{1}{4}\sqrt{(a+b+c)(b+c-a)(c+a-b)(a+b-c)} \tag{4.9}$$

Thus, the area of a triangle is approximated using the sum of the areas of its sub-triangles:

$$area(\Delta) \backsimeq \sum_{i=1}^{4^{noSub}} a_{\Delta_i} \tag{4.10}$$

Point-Surface Distance

As was said above, the distance between a triangle and a mesh is the sum of the squared distances between the sample points (obtained by *1:4* subdivisions) on the triangle and the original mesh.
$d(p_{ij}, \mathcal{M}o)$ from Eq. 4.8 is the distance from a point p_{ij} to the surface $\mathcal{M}o$.
We denote the distance between a point and a surface as the least Euclidean distance between that point and the surface:

$$d(p_i, \mathcal{M}o) = min_{p_i \in \mathcal{M}_s, q_j \in \mathcal{M}_o}(\|p_i - q_j\|) \tag{4.11}$$

For computing the Euclidean distance from a point on the simplified mesh to all points on the original mesh in order to find the least distance is time-consuming for a complex mesh.
For a mesh with n vertices, the gross computation of the distance between a point and that mesh has $O(n)$ complexity.
Therefore, for improving the time for the distance between a point and a surface, we use the modified code from the public-domain library, *Proximity Query Package* (PQP) [Tang *et al.*, 2009], [Larsen *et al.*, 1999], [Gottschalk *et al.*, 1996], [Larsen *et al.*, 2000].

Proximity Query Package

The Proximity Query Package (PQP) is a library which permits distance computation, collision detection and tolerance verification for polygonal meshes.
The main idea in computing the least distance between two objects is using a hierarchy of volumes as data structure, each volume bounding a subset of polygons (Figure 4.14).

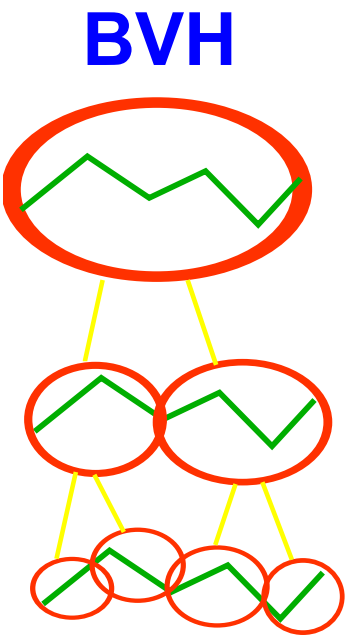


Figure 4.14: *The Bounding Volume Hierarchies used by PQP to compute the distance between two objects.*

The tree hierarchy is called *bounding volume hierarchy* (BVH) and each node is called a *bounding volume* (BV) because it bounds or contains primitives of the mesh. The root node contains the whole mesh, the leaf nodes containing, usually, one primitive.
For computing the lowest distance between two BVHs, the separation distance is initially set to infinite and the queriy starts with the root node. Recursevely, the BV's nodes are compared and if the distance is larger than the separation distance, these BVs are removed from future queries. Otherwise, the query is applied recursively to its children up to the leaf nodes.
Within the leaf nodes, the distance between the primitives is exactly computed .
The cost for computing the distance between two objects is given by the following equation:

$$C = N_{BV} \times C_{BV} + N_P \times C_P \tag{4.12}$$

where:

- N_{BV} is the number of distance tests between BVs
- C_{BV} is the cost for computing the distance for each BV
- N_P is the number of tests for computing the distances between polygons
- C_P is the cost for computing the distances between polygons

An important aspect of the distance computation queries's performance is how the nodes are defined.
In PQP, the authors introduce as BV the Rectangle Swept Sphere (RSS) which produces

fast distance queries. With this approach, each node is equivalent to a 3D rectangle swept by a sphere.
For computing the distance between two meshes using PQP, two aspects are important: the distance between two BVs and the distance between two primitives.
Using RRSs as BVs, the advantage is efficiency in calculating the distance between two BVs. This distance is computed as the distance between two 3D rectangles minus the sum of the sphere's radii.

The distance between two rectangles is determined by computing the distances between the edges pairs.
For determining the closest points of two edges, the external Voronoi regions are used. The external Voronoi region of an edge e_1 is the space outside the rectangle, which contains the points closer to e_1 than to the other element of the rectangle.
For determining the closest points between two rectangles, 16 edge pairs are checked. Two points $a \in e_1$ and $b \in e_2$ are the closest points if a is in the external Voronoi of e_2 and b is in the external Voronoi of e_1.
Based on this lemma, both distances between BVs and the distances between primitives are computed.

We use PQP in our simplification algorithm for computing the distance between a point and a mesh. Because the inputs of PQP are polygonal models, we modify the library and simulate a point as being a triangle with the vertices having the same coordinates, $v \rightarrow (v_1, v_2, v_3)$ with coordinates $p_1 = p_2 = p_3$.
For computing the distance between two meshes, the PQP library uses as input the meshes and builds the bounding volume hierarchy. Building the BVH is the most time-consuming part of PQP.
The PQP is not a dynamic library, which means that, each time when the mesh is changed, the BVH has to be rebuild.
This is a drawback for the PQP library.
In our simplification algorithm, the BVH for the simplified mesh has to be rebuilt after each edge collapse simulation and this reconstruction of the BVH is time-consuming.
Moreover, for each point for which we are computing the distance, we build a BVH model. But this one is extremly fast.

We choose to use PQP library for computing the distance from a point to a surface because it is a public software and it is fast enough.

After the presentation of the tools which are used for computing the distance between

a triangle and a surface, the distance for a triangle Δ is computed as follows:

$$d(\Delta, \mathcal{M}_o) = \sum_{i=1}^{4^{noSub}} a_{\Delta_i} \cdot d(\Delta_i, \mathcal{M}_o)^2 \tag{4.13}$$

where:

- 4^{noSub} is the number of sub-triangles of Δ
- a_{Δ_i} is the area of a sub-triangle computed using Eqn. 4.9.
- d_{Δ_i} is the distance for a sub-triangle, computed using Eqn. 4.7.

We choose as error metric the sum of the squared distances between two meshes and not the maximum of the distances in order to achieve a better evaluation of the geometric deviation between the surfaces. The evaluation is especially accurate for surfaces with features such as creases, where the estimation of the geometric deviation with the maximum distance leads to overestimating the real distance.
We use as weighted factor the area of the triangles. In this way, the distance of the triangles with a large surface has a larger weight in the value of the error metric.

4.4 Second Error Metric used: Symmetric Measure of Quadratic Error

In this section, we propose a symmetric measurement of the geometric deviation between two meshes.
The **S**ymmetric **M**easure of **Q**uadratic **E**rror denoted by **SMQE** is composed by two terms: **AMQE** (which is the distance from the simplified mesh to the original one) (see Section 4.3) and a *Reverse Error* which is the same as **AMQE** but computed from the original mesh to the simplified one. Thus, **SMQE** is the sum of these two distances:

$$\mathbf{SMQE} = \mathbf{AMQE} + ReverseError \tag{4.14}$$

We choose a symmetric measurement for the deviation introduced by an edge collapse because it provides a more accurate evaluation of the introduced error.
There are some situations such as in Figure 4.15 in which computing the distances only in one direction can lead to underestimating the deviation between two meshes.

Reverse Error

The reverse error is computed using the formula from Eqn. 4.2, but in a reverse sense, from $\mathcal{M}_o$ to $\mathcal{M}_s$. Thus, we denote the original mesh by the *source mesh*. The *destination mesh* is the simplified mesh $\mathcal{M}_s$ but with the edge collapsed (the edge for which we run

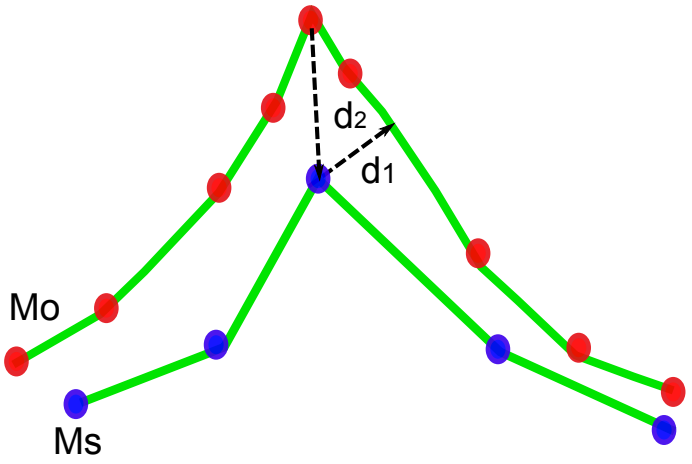


Figure 4.15: A 2D illustration of the symmetric distance between two surfaces. In this example, $d_1 < d_2$. Therefore, the one-sided distance (the distance computed from $\mathcal{M}_s$ to $\mathcal{M}_o$) leads to an underestimation of the distance value.

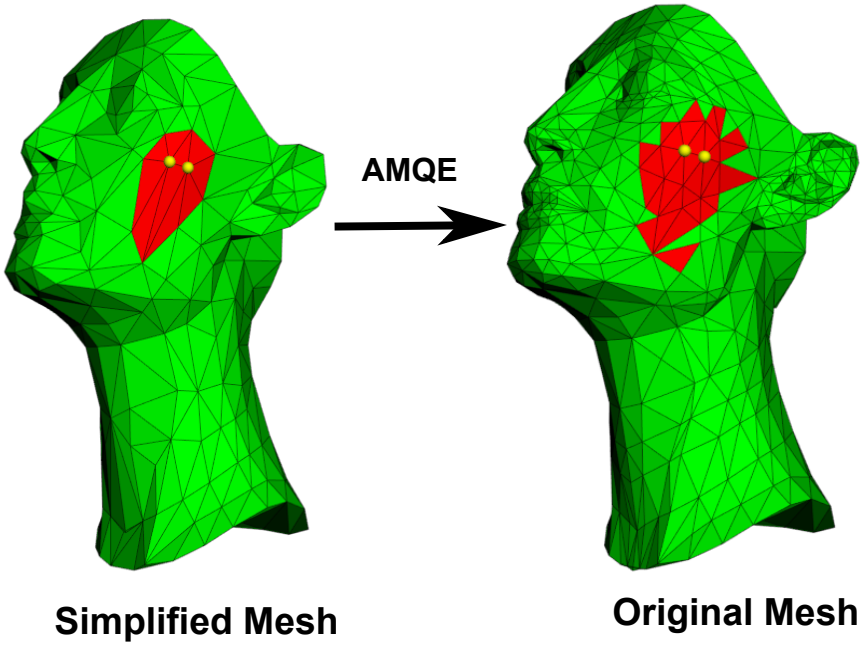


Figure 4.16: *The local direct error between two meshes.*In order to simulate the collapse for the edge with highlited endpoints, the distance is computed for all the triangles which share the endpoints (the red triangles of the Simplified Mesh). The red triangles of the Original Mesh are the triangles which are the closest of the red triangles of the Simplified Mesh.

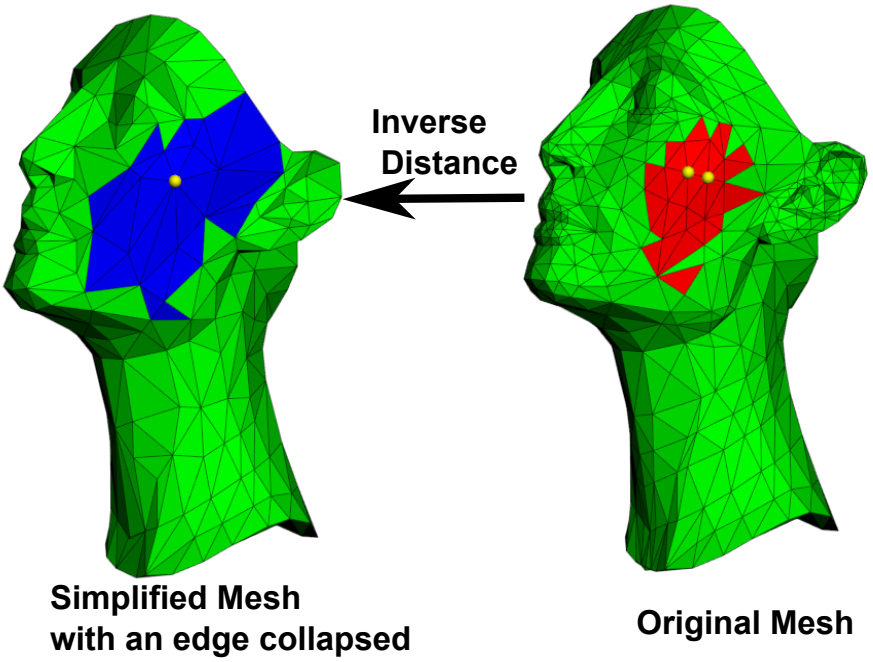


Figure 4.17: *The local reverse error between two meshes.* The red triangles of the Original Mesh are the triangles for which the distances to the Simplified Mesh are computed.

the simulation).
The reverse error is computed between the source mesh and the destination.
Therefore, when an edge collapse is simulated, for the computation of the reverse error, the destination mesh is modified.
For a rigorous computation of the reverse error introduced by an edge collapse, the triangle-area weighted sum of the squared distances has to be computed from all the triangles of the original mesh.
This approach is undesirable because of two reasons: it is time consuming to compute the distances between all triangles of the original mesh and the destination mesh for all edge collapses and for each edge collapse simulation, the destination mesh (the simplified mesh with one edge collapsed) must be recreated.
Reconstructing the destination mesh is time consuming especially for the first levels of simplification.
Moreso, each destination mesh reconstruction requires the reconstruction of the BVH structure in the PQP library. This reconstruction is time consuming.
In order to avoid these inconvenients, we propose some approximations.

First, we introduce the following notations:

- $projectionPreKernel$ (see Figure 4.16) the soup of the triangles of $\mathcal{M}_o$ which are closest to the triangles of $preKernel(\mathbf{e})$.

- $extendedPostKernel(\mathbf{v})$ (red triangles of *Original Mesh* from the Figure 4.17) is the

region which contains n-rings neighbours triangles around the $postKernel$.

The number of rings around the $postKernel$ is dependent on the stage of the simplification.
For the first steps of the simplification, when the difference in terms of complexity between $\mathcal{M}_o$ and $\mathcal{M}_s$ are small, the $extendedPostKernel$ has a number of triangles similiar to the number of triangles from the $preKernel$. For advanced simplifications, the number of triangles is smaller than in the $postKernel$. In our algorithm, the number of rings is empirically chosen to 4.

The first approximation we make is based on the idea that when a region of the simplified mesh is modified ($preKernel(\boldsymbol{e}) \rightarrow postKernel(\boldsymbol{v})$), the distance between the $projectionPreKernel$ and the $extendedPostKernel(\mathbf{v})$ is alter to a greater degree than the distances between other regions of the original mesh and the simplified mesh. Thus, in order to alterate to compute the reverse distance, we have to recompute only the distances from the $projectionPreKernel$ to the $extendedPostKernel(\mathbf{v})$.
Based on the obeservation, when a collapsed edge is simulated, for the modified region (all modified triangles), the algorithm identifies the triangles of the original mesh which are closest to the modified triangles of the simplified mesh.
In Figure 4.16, the simplified mesh has half the number of vertices of the original mesh.
Instead of recomputing the distance for all the triangles of the source mesh to the destination we only recompute the distance for the triangles of $projectionPreKernel$.
The distances for the rest of the triangles are considered to remain from the previous simplification step.
Using this approximation we recompute only for several triangles (25 triangles in the Figure 4.17 instead of 1416 triangles). This way, calculation time is reduced by 56 times.

The second approximation we make is the destination mesh in the calculation of the $ReverseError$. Ideally, when an edge collapse is simulated, a copy of the $\mathcal{M}_s$ should be created and the edge collapsed is simulated. This is time consuming, so we make the assumption that the destination mesh for the reverse error is the n-ring of the edge $\mathbf{e}$ for which we simulate the collapse.
We can consider the destination mesh as an extended $PostKernel(\mathbf{v})$. We call $extendedPostKernel$ a $PostKernel$ with several rings of neighboring triangles.
Thus, we must rebuild only the $extendedPostKernel$, which has fewer triangles than the whole mesh, $\mathcal{M}_s$, and to recompute the distances from the $projectionPreKernel$ to the $extendedPostKernel$ (Figure 4.17).

We base this approximation on the assumption that if the $preKernel(\mathbf{e})$ is projected through the direct error on a region of the original mesh, $projectionPreKernel$, this region is projected through the reverse error on a region no further than several rings of the

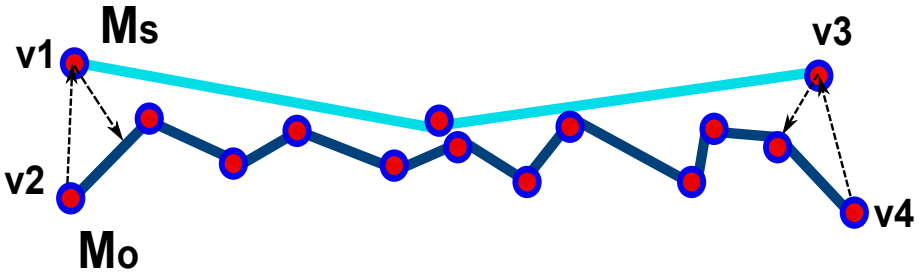


Figure 4.18: *A 2D example for application of SMQE on boundaries.*

preKernel.
By using this approximation, the destination mesh for the reverse error is reduced to several triangles. Moreover, PQP time for creating the BVH of the destination mesh is reduced.
By using the above approximations, the differences between the error computed in one direction and the error computed using the symmetric distance are not very sensitive in terms of numerical results.
The advantages of using the symmetrically measured quadratic error consist in a better preservation of boundaries and a better preservation of the features.

Boundary Preservation

In Figure 4.18, if the error is computed only in one direction, from $\mathcal{M}_s$ to $\mathcal{M}_o$, for the vertices on the boundaries, as v_1 and v_3, the distances to the $\mathcal{M}_o$ are underestimated. In the end, one of the boundary edges can be selected to be collapsed. Therefore, the boundaries are not preserved using only the direct error.
When the reverse error (the distance from $\mathcal{M}_o$ to $\mathcal{M}_s$) is involved in the error estimation, we can see the distances from the boundary vertices v_2 and v_4 to the simplified mesh, $\mathcal{M}_s$ can be larger than the direct error. Therefore, the boundary edges are preserved.

Feature and Detail Preservation

Another advantage of using SMQE as a measure for evaluating the geometric error introduced by an edge collapse is the preservation of features.
For example, for two meshes as in Figure 4.19, if we compute only the distances from $\mathcal{M}_s$ to $\mathcal{M}_o$, because d_1 has a small value, the crease of $\mathcal{M}_s$ can be eliminated. If we compute the symmetric distance, the distance from $\mathcal{M}_o$ to $\mathcal{M}_s$, because $d_2 >> d_1$, the edges which form the crease will not be picked for collapse.

4.5 QEM-based Vertex Optimization

When an edge is collapsed, the endpoints of the edge are merged into a single vertex:

$$\mathbf{e} = (v_1, v_2) \rightarrow \bar{v} \tag{4.15}$$

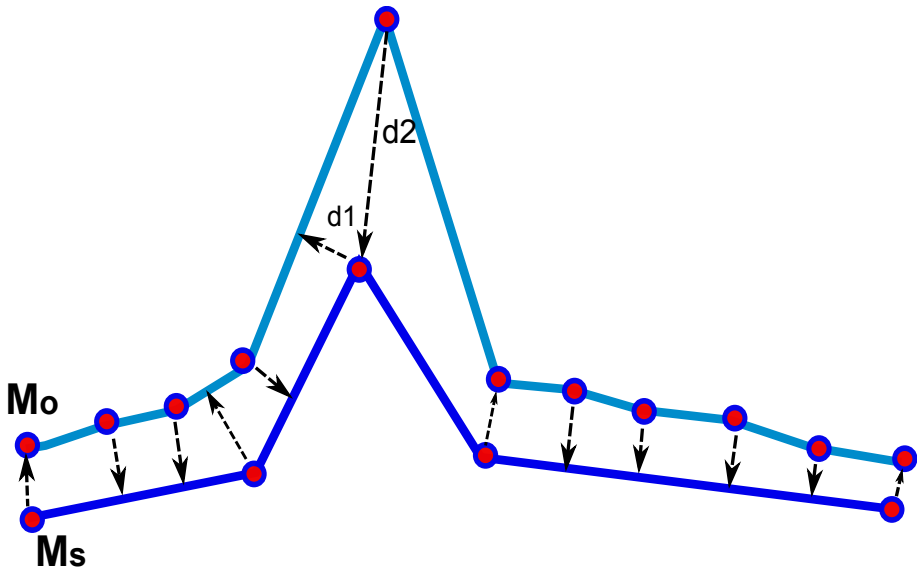


Figure 4.19: *A 2D example for importance of SMQE in the preservation of the features of a mesh.*

We call this vertex the *resulting vertex*.
The position of the resulting vertex, $\bar{v}$ is very important in the quality of the simplified mesh.
The position of $\bar{v}$ has to be chosen so that it approximates best the original mesh.
Thus, we must define how to best place the resulting vertex, $\bar{v}$.
To find the position of the resulting vertex, we follow [Garland and Heckbert, 1997].
Thus, each vertex in the original mesh is considered to be the solution of the planes of faces surrounding it.
When an edge is collapsed to a vertex, the distance from the resulted vertex to the planes surrounding it is considered to be the cost of the edge collapse.
In [Garland and Heckbert, 1997], each face in the original mesh has associated a quadratic matrix, Q.
If we have a plane $p = (n, d)$ where $n = (n_x, n_y, n_z)$ is the normal and d is the displacement of the plane to the origin and a point $v = (x, y, z)$ which belongs to the plane, the equation of the plane is:

$$\mathbf{p} : x \cdot n_x + y \cdot n_y + z \cdot n_z + d = 0 \tag{4.16}$$

with

$$n_x^2 + n_y^2 + n_z^2 = 1 \tag{4.17}$$

If we write $p = [n_x, n_y, n_z, d]$ and $v = [x, y, z, 1]$,thus, the plane equation can be written as:

$$v \cdot p^T = 0 \tag{4.18}$$

When v is moved away from the plane, we denote its new position by $\bar{v}$, the quadratic distance between $\bar{v}$ and the plane, p is:

$$\begin{aligned}\Delta_p(\bar{v}) =& (vp^T)^2 \\ &= (vp^T)\cdot(pv^T) = v\cdot(p^Tp)\cdot v^T \\ &= v\cdot K_p\cdot v^T\end{aligned} \tag{4.19}$$

$$K_p = \begin{vmatrix} n_x^2 & n_xn_y & n_xn_z & n_xd \\ n_xn_y & n_y^2 & n_yn_z & n_yd \\ n_xn_z & n_yn_z & n_z^2 & n_zd \\ n_xd & n_yd & n_zd & d^2 \end{vmatrix} \tag{4.20}$$

The sum of the quadratic distances from the point v to the set of the planes associated to v is:

$$\begin{aligned}\Delta(v) =& \sum_{p\in Planes(v)} \Delta_p(v) \\ &= \sum_{p\in Planes(v)} (v\cdot K_p\cdot v^T) \\ &= v\cdot(\sum_{p\in Planes(v)} K_p)\cdot v^T \\ &= v\cdot Q_v\cdot v^T\end{aligned} \tag{4.21}$$

Thus, Q_v is the quadratic error associated to the vertex v. After an edge contraction, $\mathbf{e} = (v1, v2) \rightarrow \bar{v}$, the resulted vertex inherits the quadratics of the endpoints of the collapsed edge:

$$Q_{\bar{v}} \leftarrow Q_{v_1} + Q_{v_2} \tag{4.22}$$

Because each resulted vertex inherits the planes surrounding the endpoints of the collapsed edge, the original planes are preserved during the simplification. Thus, the error introduced by an edge collapse is computed with reference to the original mesh and not to the previous simplified mesh.

One solution for the position of the resulted vertex is one of its endpoints: $\bar{v} = v_1$ or $\bar{v} = v_2$.

For each possibility, $Q(\bar{v})$ is computed and the position which produces the smallest value from $Q(v_1)$ and $Q(v_2)$ is picked.

By using this approach, the set of vertices of the simplified mesh is the subset of the vertices of the original one (v_1 or v_2). For this reason, the present approach does not produce the best approximations.

Because the sum of the squared distances from the resulting vertex to the set of planes is quadratic, the sum has a minimum (Figure 4.20).

Therefore, the minimum of this sum is the optimal resulting vertex position, from the

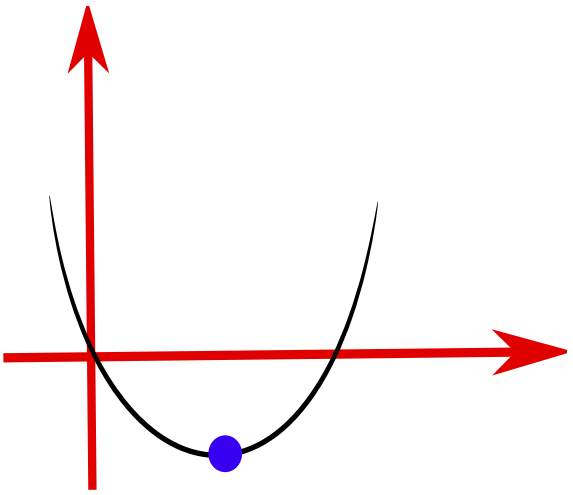


Figure 4.20: *The quadratic form of the sum of squared distances from a point to its associated set of planes.*

quadratic error point of view.
The resulting vertex minimizes the sum of the squared distances from this vertex to the set of planes surrounding it.
Thus, the resulting vertex, $\bar{v}$ minimizes the sum $\Delta(\bar{v})$:

$$\frac{\partial \Delta}{\partial x} = \frac{\partial \Delta}{\partial y} = \frac{\partial \Delta}{\partial z} = 0 \tag{4.23}$$

The resulting vertex position is the solution to the system of linear normal equations:

$$\begin{vmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{vmatrix} \cdot v = \begin{vmatrix} 0 \\ 0 \\ 0 \\ 1 \end{vmatrix} \tag{4.24}$$

And

$$v = \begin{vmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{vmatrix}^{-1} \cdot \begin{vmatrix} 0 \\ 0 \\ 0 \\ 1 \end{vmatrix} \tag{4.25}$$

Because the distance from a point to its surrounding planes is written as:

$$Q(v) = v^T A v + 2b^T v + c \tag{4.26}$$

By solving $\Delta Q(v) = 0$, the resulting vertex position is:

$$\bar{v} = -A^{-1} b \tag{4.27}$$

In the Eqn. 4.27, the condition for getting the vertex position is that A must be an invertible matrix.
Theoretically, if A is singular, it does not have an inverse. So, the Eqn. 4.27 cannot be solved.
A matrix is singular if its determinant, $det(A) = 0$.
If A is not singular, then the solution of Eqn. 4.27 will be an infinity of points which form a line or a plane. This situation occurs when all the planes surrounding the vertex for which we compute the positions are parallel.

In the situation that the quadratic matrix is not invertible, we follow the approach proposed in [Lindstrom, 2000].
For the matrix A, the *singular value decomposition* is performed: $A = U\Sigma V^T$.
Thus, the optimal new vertex position becomes:

$$\bar{v} = \hat{v} + V\Sigma^+ U^T(b - A\hat{v}) \tag{4.28}$$

where:

- $\hat{v}$ is the midpoint of the collapsed edge: $\hat{v} = \frac{v_1+v_2}{2}$
- U, Σ, V are the matrices obtained from *sigular value decomposition*

When $\Sigma^+ = \Sigma$, the Eqn. 4.28 is reduced to $A^{-1}b$.

We choose to compute the position of the new vertex by using the minimization of QEM (Eqn. 4.24) because of simplicity of computation. In order to compute the new vertex position it is enough to compute the quadratic matrix for each vertex.
The resulting vertex is the one which minimizes the QEM between that vertex and the planes surrounding it. This supposes the resulting vertex fits the geometry of the original mesh.
This represents an advantage.

The disadvantage is that the quadratic error computes the distance from the resulting vertex to the supporting planes of its surrounding triangles and not directly to the triangles. For curved meshes, the real distance is underestimated and leads to a poor approximation of the true error introduced by an edge collapse. Moreover, the position of the new vertex is poorly computed, therefore, the simplified mesh does not fit the geometry of the original mesh very well.

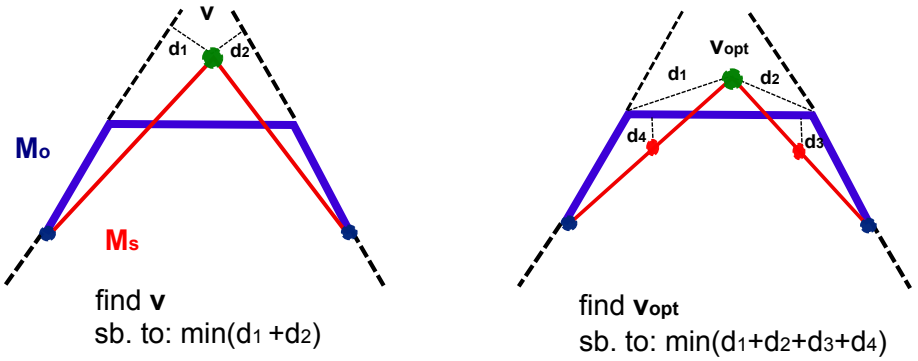


Figure 4.21: *The 2D representation of the vertex placement.* In the left image, the new vertex is placed to minimize the sum of quadratic errors to the planes. In the right image, the vertex minimizes the sum of squared distances to its adjacent faces.

4.6 Volume-based Vertex Optimization

The position of the resulted vertex $\bar{v}$ obtained with the Eqn. 4.28 is optimal only in terms of the quadratic error metric. Because the quadratic distances are computed from the resulting vertex to the planes of its surrounding faces, for curved surfaces, the resulted vertex position is not optimal.
Thus, the new vertex can be placed on an optimal position which accurately approximates the geometry of the original mesh.
Therefore, we introduce in the simplification algorithm a post-processing step which iteratively moves the position of the resulted vertex to an optimal one.
Therefore, the simplification algorithm has a new step:

1. On the original mesh, all possible edge collapses are simulated.
2. The edge which introduces by its collapse the least geometric deviation is chosen for collapse.
3. The edge is collapsed to a single vertex $e = (v_1, v_2) \rightarrow \bar{v}$. The following operations are performed:
   - the position of the resulted vertex, $\bar{v}$ is computed using Eqn. 4.28: $(v_1, v_2 \rightarrow \bar{v})$.
   - improve the position of $\bar{v}$: $\bar{v} \rightarrow \bar{v}_{opt}$
   - all the faces which previously were connected to v_1 and v_2 are now connected to $\bar{v}$.
   - the new vertex, $\bar{v}$ takes the id v_1.
   - the vertex v_2 and degenerated faces (the faces which shared the collapsed edge) are eliminated;
   - the errors are reevaluated for all edges of the simplified model.
4. All those steps are repeated until the stop condition is reached.

Each time when the edge collapse error is simulated, after the resulted vertex position is computed, the improvement of the vertex position is applied.
We use an optimization algorithm which improves the position of the new vertex minimizing the volume embedded between the simplified mesh and the original one.
For simplificity, the volume is computed between the $postKernel(v)$ and $\mathcal{M}_o$.
In Figure 4.21, on the left, the position of the resulting vertex is chosen in order to minimize the sum of the distances to the planes of the faces of the original mesh. The distances from the resulted vertex to the planes are not identical to the distances from the vertex to the faces of the original mesh, as in Figure 4.21, right.
Therefore, the resulting vertex does not fit very accurately with the geometry of the original mesh.
We propose a pre-processing step, which improves the position of the resulting vertex, and starts with the actual position.
For this purpose, distances are computed from the sample points of the $preKernel(v)$ to the $\mathcal{M}_o$. The vertex v is moved with a displacement given by the sum of these distances. The process is iterative, and after n iterations, the distance becomes close to 0. The resulting position for v becomes the *optimal position*, v_{opt}.
For expressing the volume embedded between $\mathcal{M}_s$ and $\mathcal{M}_o$, we follow [Alliez *et al.*, 1999] with some modifications.
The optimal new vertex position, v_{opt} is that which minimizes the volume embedded between the simplified mesh, $\mathcal{M}_s$ and the original one, $\mathcal{M}_o$.
The optimal position is achieved by an iterative volume minimization process.

The most difficult part is the embbeded volume computation.

Following [Alliez *et al.*, 1999], we define the volume embedded between two meshes $\mathcal{M}_o$ and $\mathcal{M}_s$ by:

$$V(\mathcal{M}_o, \mathcal{M}_s) = \int \int_{uv} \vec{v}(\mathcal{M}_o, \mathcal{M}_s) d\sigma(u, v) \tag{4.29}$$

where

- $\vec{v}(\mathcal{M}_o, \mathcal{M}_s) d\sigma(u, v)$ is the elementary volume between $\mathcal{M}_o$ and $\mathcal{M}_s$
- $\vec{v}(\mathcal{M}_o, \mathcal{M}_s)$ represents the distance vector from $\mathcal{M}_o$ to $\mathcal{M}_s$ on the surface patch $dudv$
- $d\sigma(u, v)$ is the patch area

The distance vector $\vec{v}$ takes into account the dependency of the two meshes.
The distance vector is negative if it has the same orientation as the normal $n(u, v)$ to the mesh $\mathcal{M}_s$ and positive if they have different orientations.
Since a mesh can be defined as a discrete set of vertices, we can rewrite $\mathcal{M}_s(u, v)$ as an

interpolation between its vertices:

$$V(\mathcal{M}_s, \mathcal{M}_o) = \sum_{i=1}^{N} \lambda_i(u,v) X_i \tag{4.30}$$

where:

- X_i represents the mesh vertices defined in R^3
- λ_i denotes the shape function

For a point inside a triangle in a mesh, we consider the shape function as being the linear interpolation of the triangle vertices. Now, we can express the volume between $\mathcal{M}_o$ and $\mathcal{M}_s$ as:

$$\mathcal{M}_s(u,v) = \sum_{i=1}^{N} \int\int_{uv} \vec{v}(\mathcal{M}_o, \mathcal{M}_s) \lambda_i(u,v) v_i d\sigma(u,v) \tag{4.31}$$

We discretize Eqn. 4.31 by sampling each face of the mesh. Each triangle is subdivided into a fixed number of infinitesimal sub-triangles, and under these conditions Eqn.4.31 becomes:

$$V(\mathcal{M}_o, \mathcal{M}_s(u,v)) = \sum_{i=1}^{N} \sum_{c \in Cells} \lambda_i(u,v) X_i \vec{v}(\mathcal{M}_o, \mathcal{M}_s) d\sigma(u,v) \tag{4.32}$$

In order to minimize the volume between the faces adjacent to that vertex and the original mesh, and implicitly to reduce the quadratic error between the approximated mesh and the original one, we introduce the following formula:

$$disp(X_i) = \frac{\sum_{c \in PostKernel(X_i)} \lambda_c w_c d(c, \mathcal{M}_o)}{\sum_{c \in PostKernel(X_i)} \lambda_c w_c} \tag{4.33}$$

where

- $disp(X_i)$ is the displacement for the vertex X_i
- $Supp(X_i)$ denotes the adjacent faces of X_i

The direction of the displacement is given by the vectorial sum of the distances between $\mathcal{M}_s$ and $\mathcal{M}_o$.

- if $\vec{n}\vec{v} > 0$ then $\vec{v} = -sign(\vec{v})$
- else, $\vec{v} = sign(\vec{v})$

where $\vec{n}$ is the normal unit vector on the current face.
The magnitude of the movement is given by the sum of the distances from the sub-triangles to the original model. The sum is scaled by the sub-triangle areas and by the shape function λ_c.

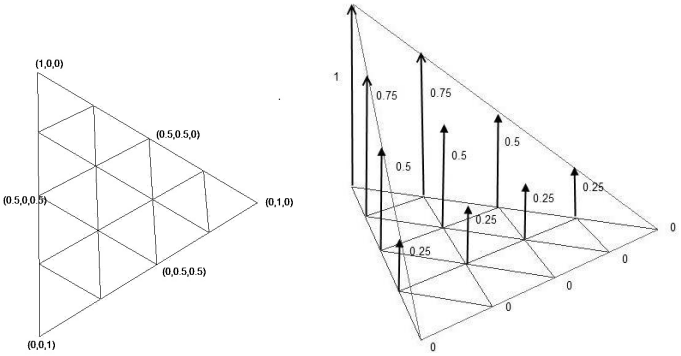


Figure 4.22: **The shape functions.** The shape function of a point inside a face is determined from the barycentric coordinates of the respective point.

λ_c is set to 1 on the X_i and decreases to 0 towards the neighbouring vertices (Figure 4.22). For a given point inside a triangle, we deduce the value of the function shape using the barycentric coordinates of the respective point. Thus, if the point has the barycentric coordinates $(\lambda_1, \lambda_2, \lambda_3)$, the shape function for the respective point will be λ_1.
The shape function of a sub-triangle is the arithmetic mean of the shape functions of its vertices.

In practice, we obtain the magnitude and the direction of the distances using the PQP library. The evolution of the vertex position, for the k^{th} step can be written as:

$$X_i^{k^{th}} = X_i^{k^{th}-1} + disp(X_i^{k^{th}-1}) \tag{4.34}$$

After each optimization step, the volume embedded between the approximated mesh and the original one is reduced. Therefore, the quadratic error and the Hausdorff distance between those meshes decrease (Figure 4.23).

In Figure 4.23 the goal is minimizing the volume embedded between two meshes: the mesh represented with the red wireframe and the green mesh. The volume between the meshes is computed using Eqn. 4.32. Using Eqn. 4.34, the vertex v is moved to the green mesh, in order to minimize the volume between meshes. The volume minimization is an iterative process. In the above example, after the 5^{th} iteration, the embedded volume becomes close to 0 (0.00312992).
The embedded volume is in the beginning 0.611 and decreases to 0 (Fig. 4.24).

Form Figure 4.24 we see that the biggest decrease (from 0.66 to 0.13) volume is after the first iteration, the decresing growing smaller while iterations increase.
Because the embedded volume is computed by using the quadratic distances between the sample points of the two meshes it is normal behaviour for the displacement of the target vertex to be larger in the beginning and decrease later.

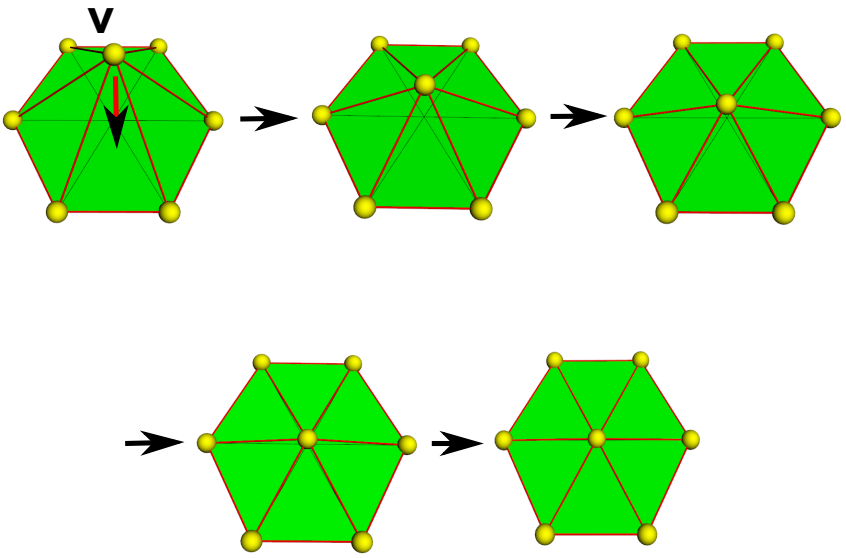


Figure 4.23: *Volume minimization between two meshes.*

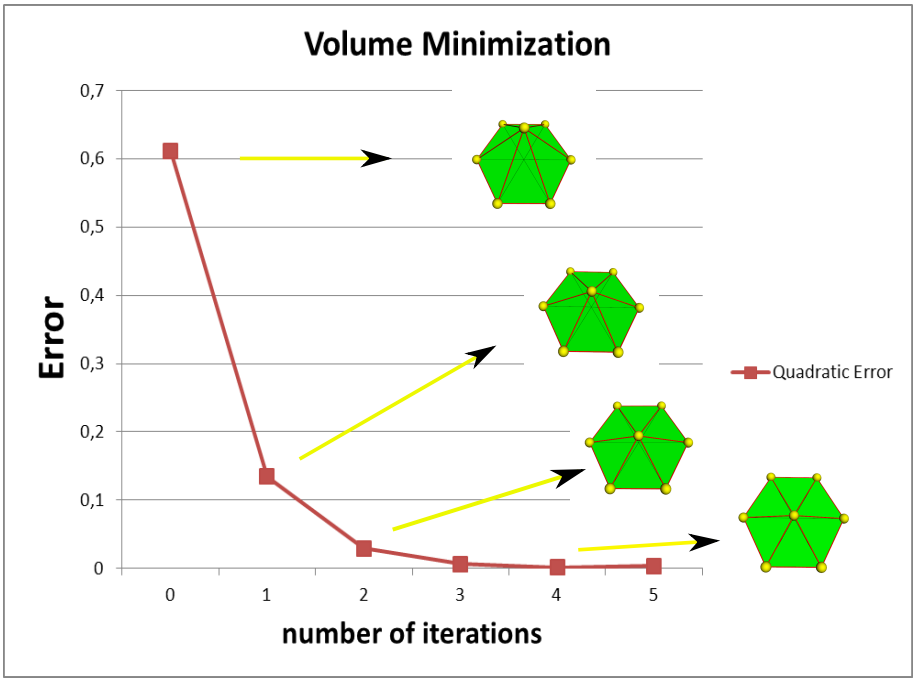


Figure 4.24: *The variation of the volume embedded between two meshes with the number of iterations.*

Chapter 5

Results

In this chapter we present the results obtained with our simplification algorithm. The simplifications are obtained using both the AMQE and the SMQE geometric error metrics and are compared with the simplifications obtained using Qudratic Error Metric (QEM) [Garland and Heckbert, 1997].
In order to evaluate the quality of the simplifications obtained using our algorithm, we use the quadratic error metric and the Hausdorff distance. The Hausdorff distance is measured using the *Metro* tool.
We will compare the results in terms of: preservation of detail, preservation of boundaries and islands and errors introduced during simplification (the errors between the simplified model and the original one).

5.1 Sharp Features Preservation

Pieta model
First we simplify the Pieta model with 13 940 vertices to a model with 900 vertices using the both the SMQE (Figure 5.1) and the QEM (Figure 5.2) methods. For the Pieta model simplified with SMQE, details such as the eyes, nose or mouth are more accurately preserved than in the model simplified with QEM.
Moreover, the waves of the Pieta moodel's kerchief are better preserved.
For the base of the statue, which is almost a plane surface, the SMQE method uses less triangles than QEM. For the same budget of vertices, SMQE uses more triangles to approximate curved regions and less for flat regions, while QEM uses more triangles for flat regions. Therefore the curved regions (such as the face of Mary) are not well preserved

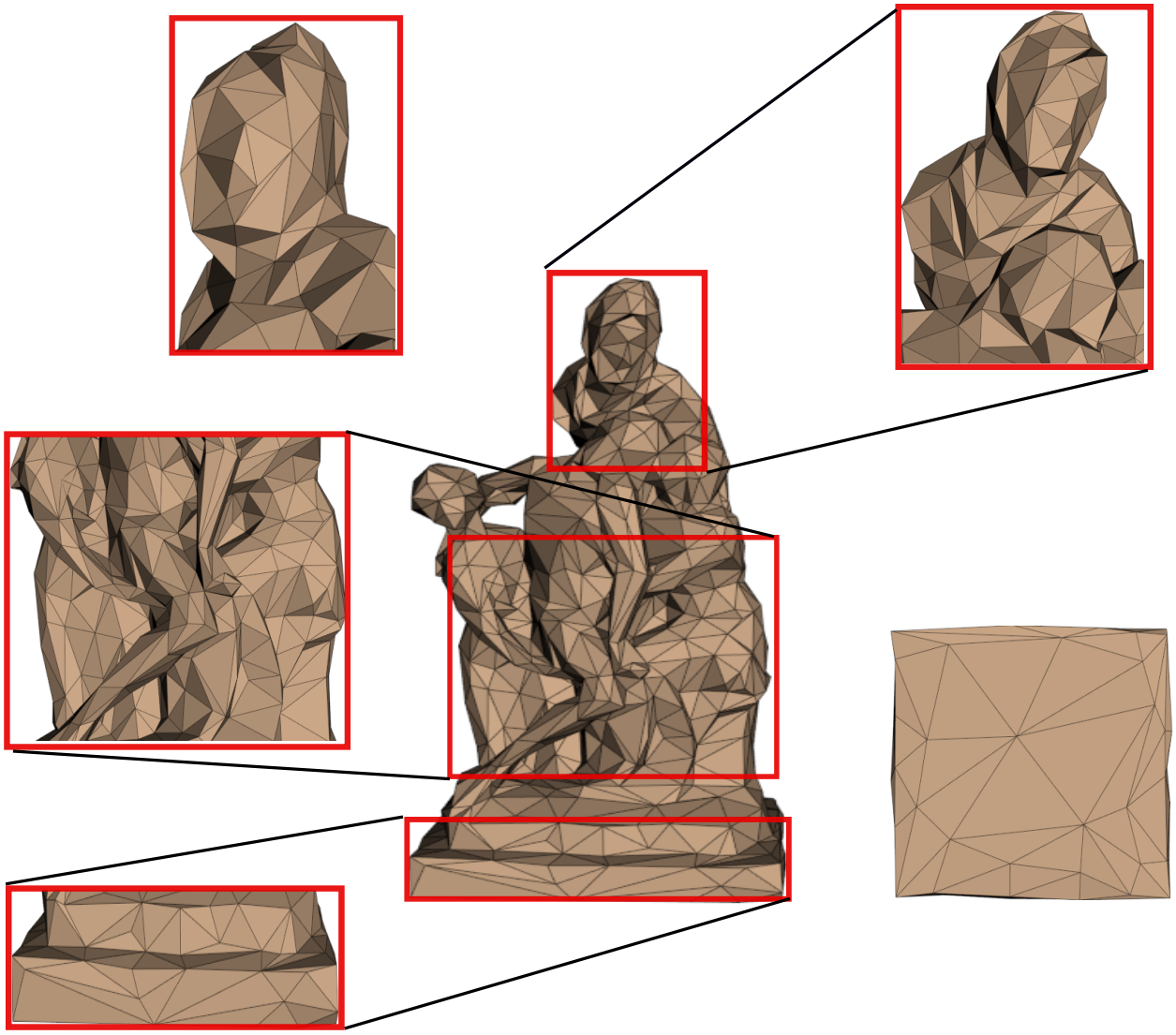


Figure 5.1: The simplified Pieta model with 900 vertices using SMQE.

with QEM.

The quadratic error for the Pieta model simplified down to 2000 vertices with SMQE (Figure 5.3, top) is 0.008878 while for Pieta model down to the same number of vertices with QEM (Figure 5.3, bottom) is 0.010346. The Hausdorff distance is 3.714012 for SMQE and 4.75883 for the model simplified with QEM.

In conclusion, both the quadratic error and the Hausdorff distance are smaller for the model simplified with SMQE than for the model simplified with QEM.

In Figure 5.2 the base of the Pieta model simplified with SMQE has fewer triangles than of the Pieta model simplified with QEM.

In the same figure, the leg of Jesus is better outlined for the Pieta model with 900 vertices simplified with SMQE than for the model with the same complexity but simplified with QEM. Moreover, the waves of Mary's dress are better outlined for SMQE than for QEM.

In Figure 5.4 is represented the quadratic error versus the number of vertices. The

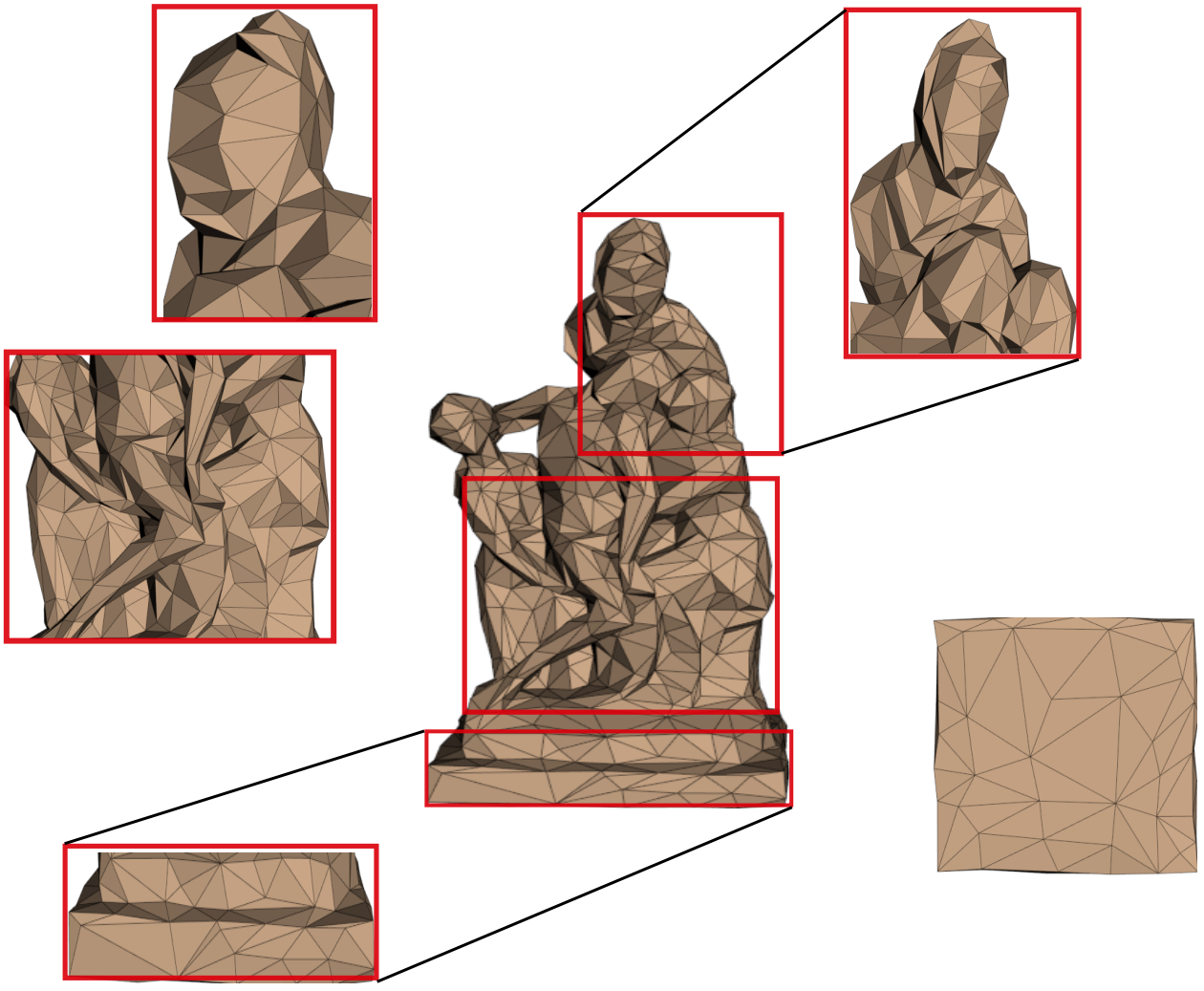


Figure 5.2: The simplified Pieta model with 900 vertices using QEM.

curve of the quadratic error between the original Pieta model and the model simplified with SMQE is always bellow the curve which represents the quadratic error introduced in simplification by QEM.
Because the differences between the model simplified with SMQE and the one simplified with QEM are relevant for extreme simplifications (less than 90% of to the original complexity) in order to improve the simplification, we simplified the original Pieta model using AMQE down to 5000 vertices and afterwards started simplification with SMQE.
In Figure 5.5, the geometric error introduced by simplification with SMQE and QEM is given by the Hausdorff distance.
We notice that the Hausdorff distance between the original and the simplified model with 5000 vertices is almost identical to the Hausdorff distance between the original and the model with 500 vertices (2.90932). This means that most geometric deviation is introduced before simplifying the mesh down to 5000 vertices. All the simplifications made from 5000 to 500 vertices do not introduce a value higher than 2.90932.
The same behaviour of the model simplified with QEM. The Hausdorff distance is 4.758183 for all simplified models between 5000 and 500 vertices.
In Figure 5.6, we display the distances between the original Pieta and the model simplified until 900 vertices using colours. The distances are computed between each vertex of the original model to the simplified one using the library PQP.

Octa-flower model: The 7919 vertices Octa-flower model (Figure 5.7) is simplified down to 169 and respectively 99 vertices. The model is simplified using SMQE, AMQE and QEM error metrics.
The shape of the model is better preserved for the model simplified with SMQE.
The spirals of the Octa-flower model are better outlined for the Octa-flower with 99 vertices simplified with both SMQE and AMQE than QEM.
The same situation is present for the model with 169 vertices.
The 99 vertices Octa-flower model is simplified with SMQE the quadratic error is 0.018293 and the Hausdorff distance is 0.78752. For the model with the same complexity but simplified with AMQE the quadratic error is 0.0209451 and the Hausdorff distance is 1.006559 and for the model simplifed with QEM, 0.03528 and 1.347396 respectively.

The difference between the quadratic error introduced into simplifications by the SMQE and AMQE is not so great (Figure 5.8), but is visually lower than the quadratic error introduced by the QEM.
The Hausdorff distance introduced by simplifications is almost identical (0.1137) for the models simplifed with SMQE and AMQE down to 459 vertices (see Figure 5.9). Down to this budget of vertices, the Hausdorff distance for the model simplifed with QEM is higher. For the models with fewer than 459 vertices, the Hausdorff distance introduced by AMQE begins to increase until the value of 0.7676 for the model with 159 vertices, being almost equal to the Hausdorff distance introduced by the QEM (0.747582 for the same number of vertices).

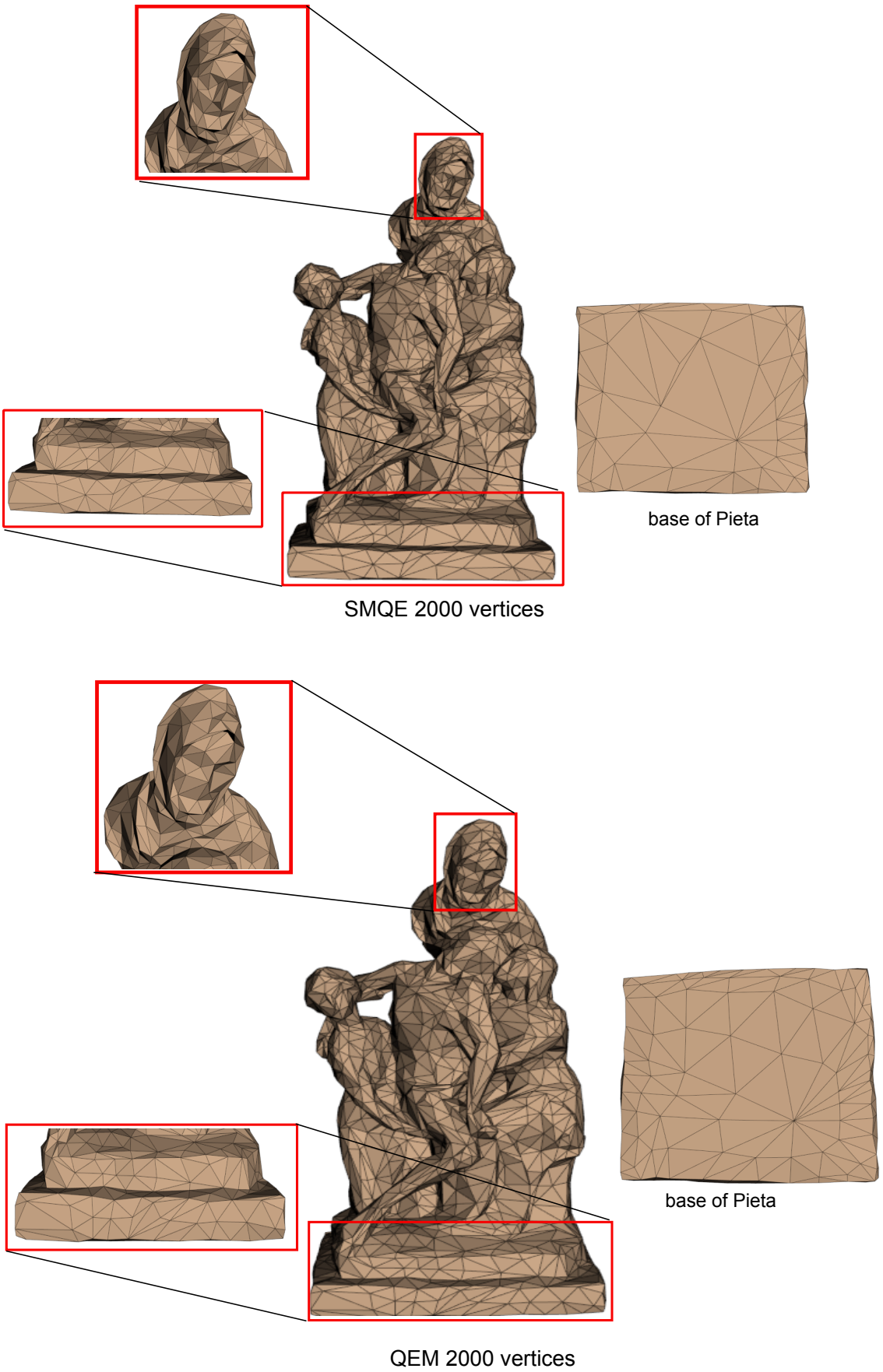


Figure 5.3: The simplified Pieta model with 2000 vertices using SMQE (top) and QEM (bottom).

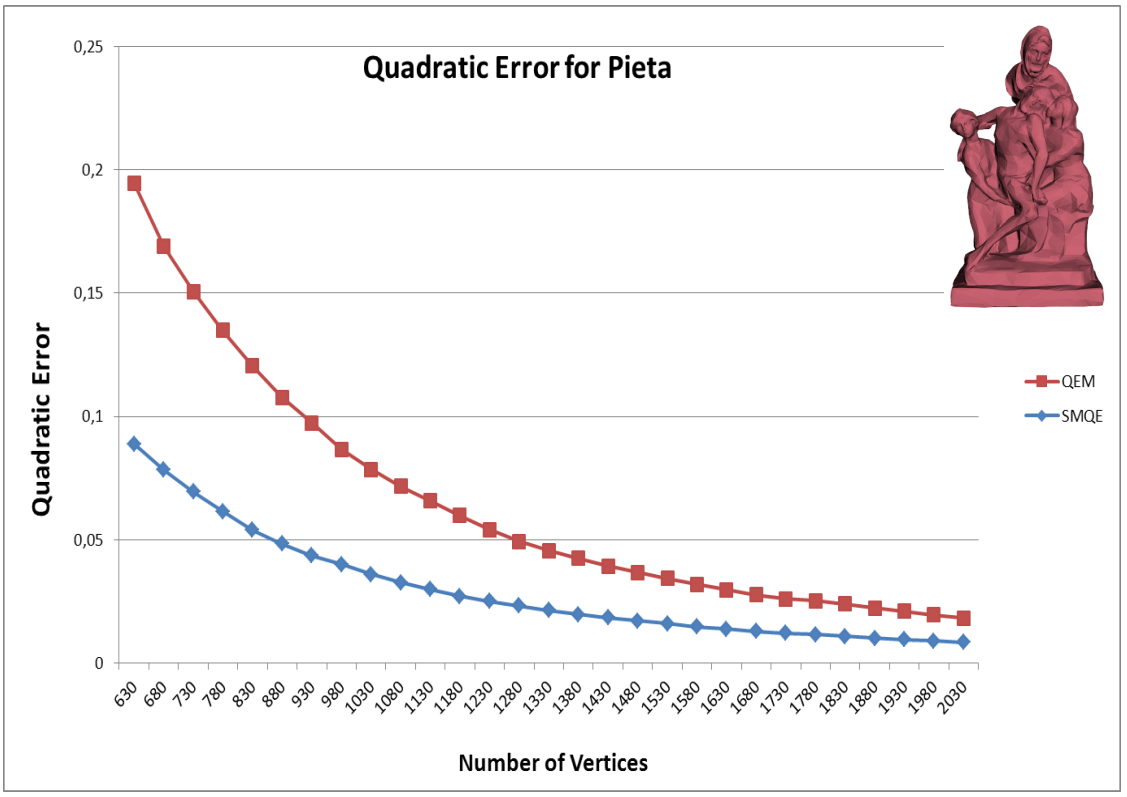


Figure 5.4: The quadratic error introduced in simplification of Pieta model.

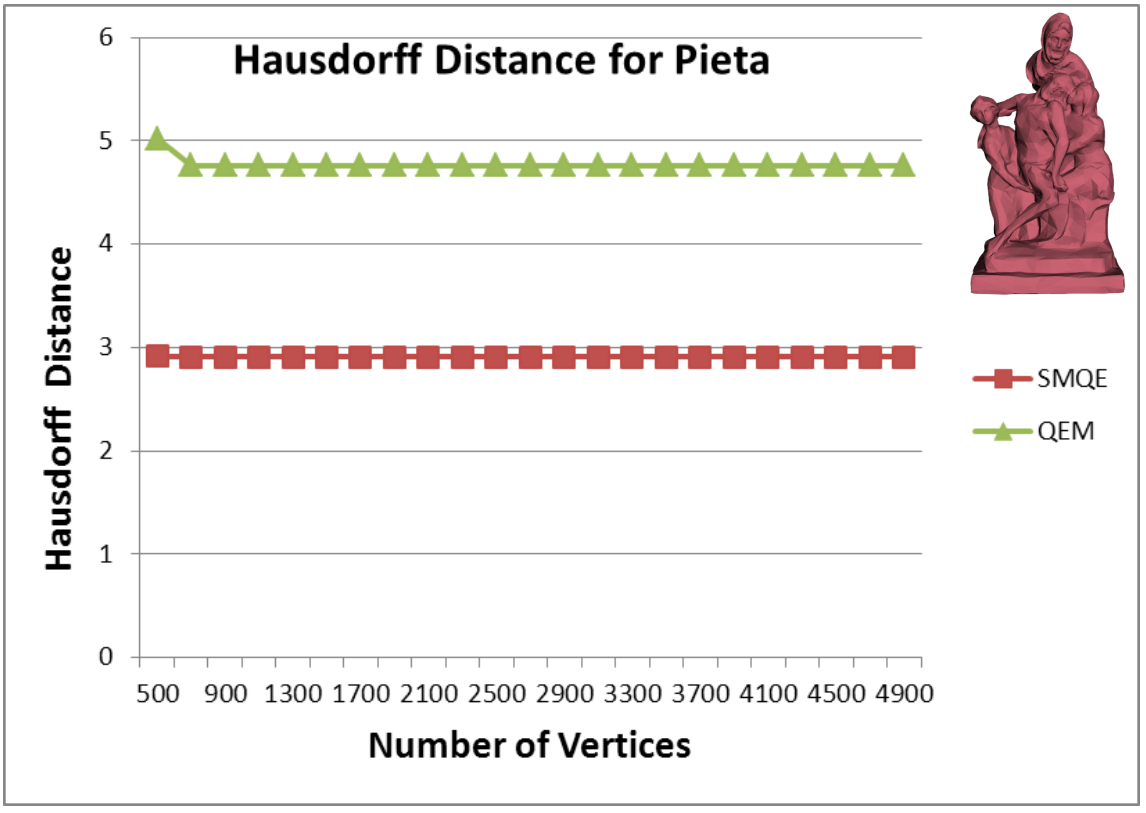


Figure 5.5: The Hausdorff distance introduced in simplification of Pieta model.

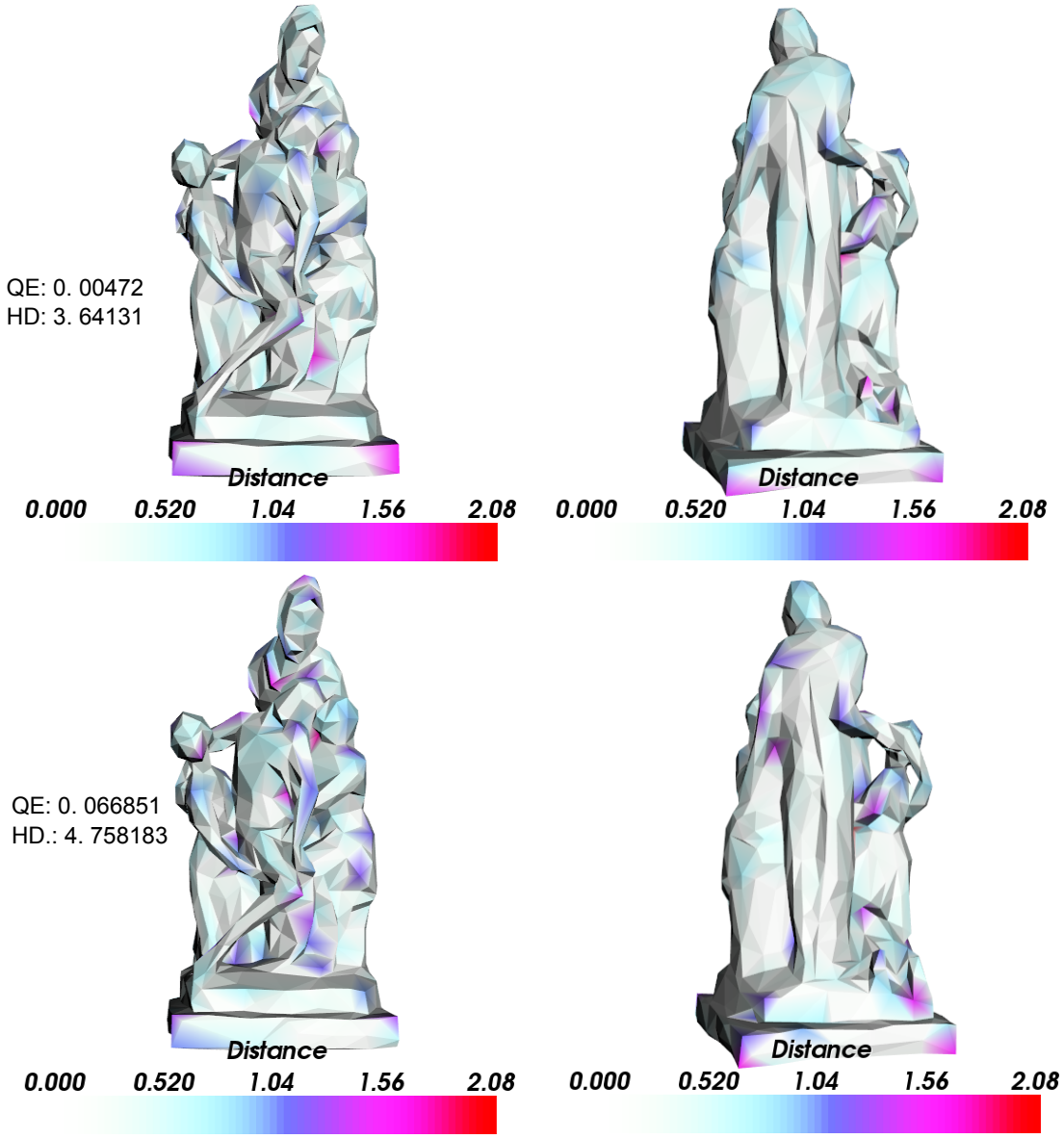


Figure 5.6: The distance between the Pieta model with 13 940 vertices and its simplified version with 900 vertices. The distance is computed between each vertex of the original model to the simplified model by using the PQP library.

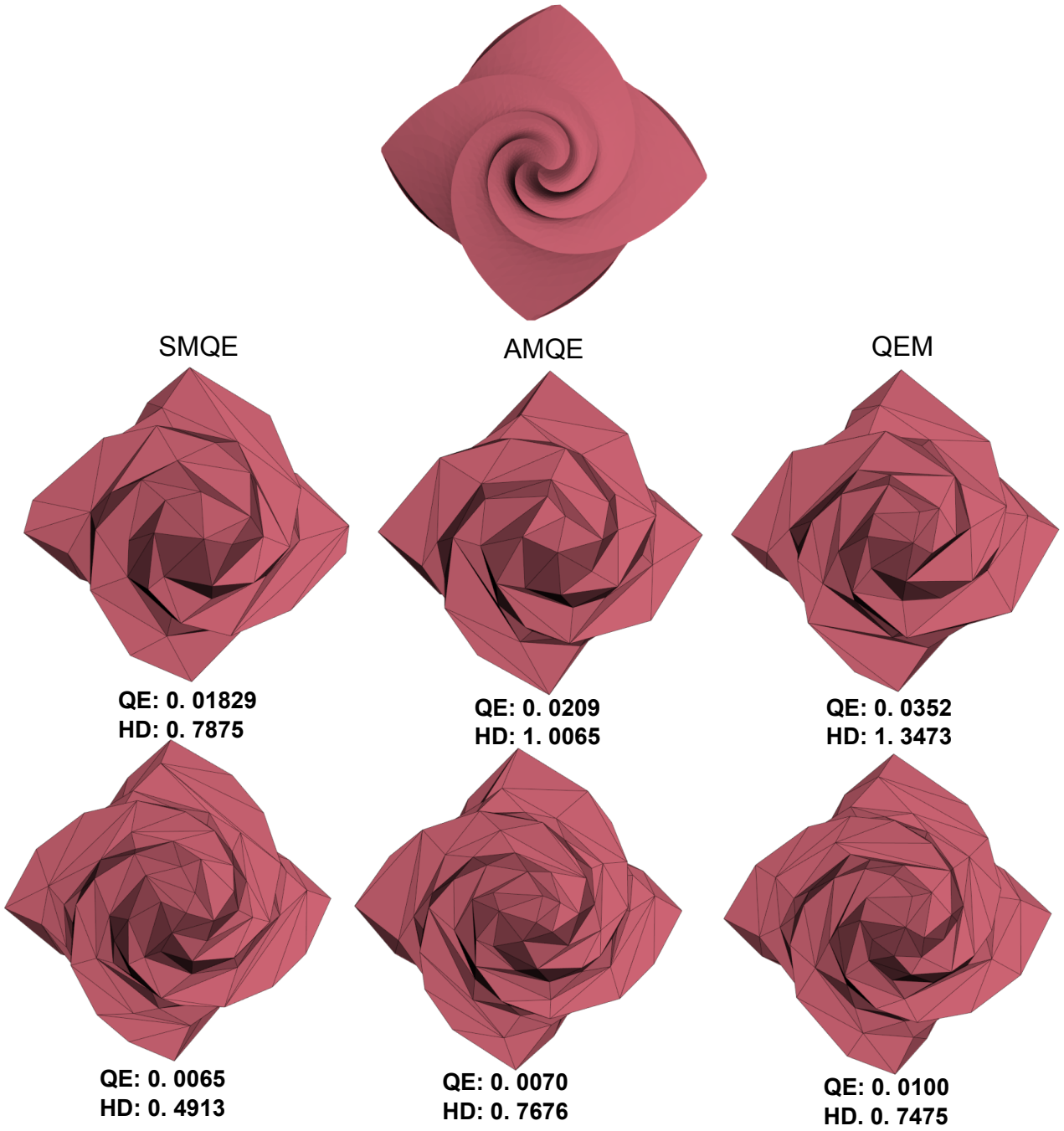


Figure 5.7: Simplifications of Octa-flower model using SMQE, AMQE and QEM. On the top, the simplified model has 99 vertices and 169 on the bottom.

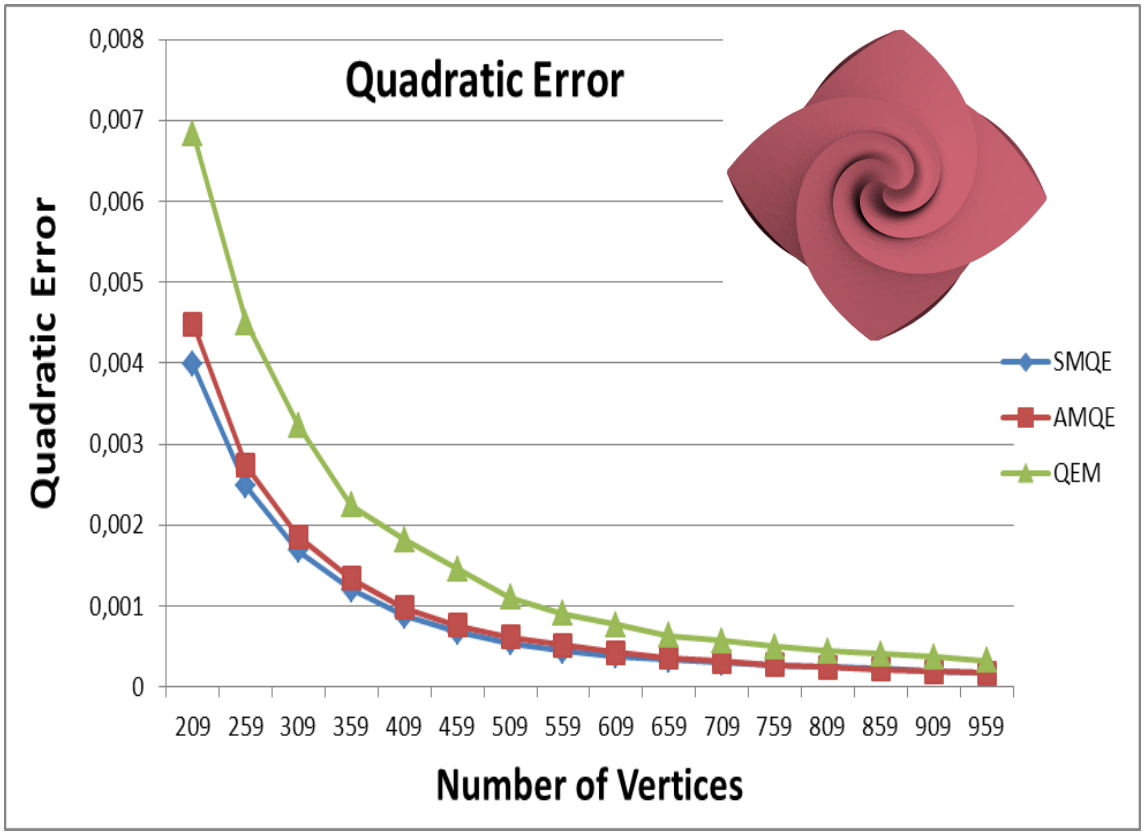


Figure 5.8: The quadratic error introduced in the simplification of the Octa-flower.

In Figure 5.10 the distances between the original Octa-flower model and the model simplified until 169 vertices using SMQE, AMQE and QEM error metrics are displayed with colurs.

Horse model: In Figure 5.11 is presented a sequence of simplifications for the Horse model (with 19 851 vertices and 39698 trinagles). The model is simplified using SMQE, AMQE and QEM. The simplified models have 801 vertices and 1595 triangles, 411 vertices and 816 triangles and 161 vertices and 316 triangles.
From Figure 5.12 we can see that the details are better preserved by using SMQE than with the other methods.
The ears are preserved for the 161 vertices Horse model simplified with SMQE (Figure 5.12, first column). For the Horse with the same number of vertices but simplified with AMQE, just one ear is preserved, both ears being simplified for the Horse model simplified with QEM.
The muzzle of the horse is better preserved with SMQE and less well preserved with AMQE and QEM (Figure 5.12, left column).
The left rear leg hoof (Figure 5.12, the right column) is better reproduced for Horse model simplified with SMQE. The hoof is least preserved for the model simplified with QEM.
The fetlock and the hock are more faithful to the originals for the models simplified with SMQE and AMQE.

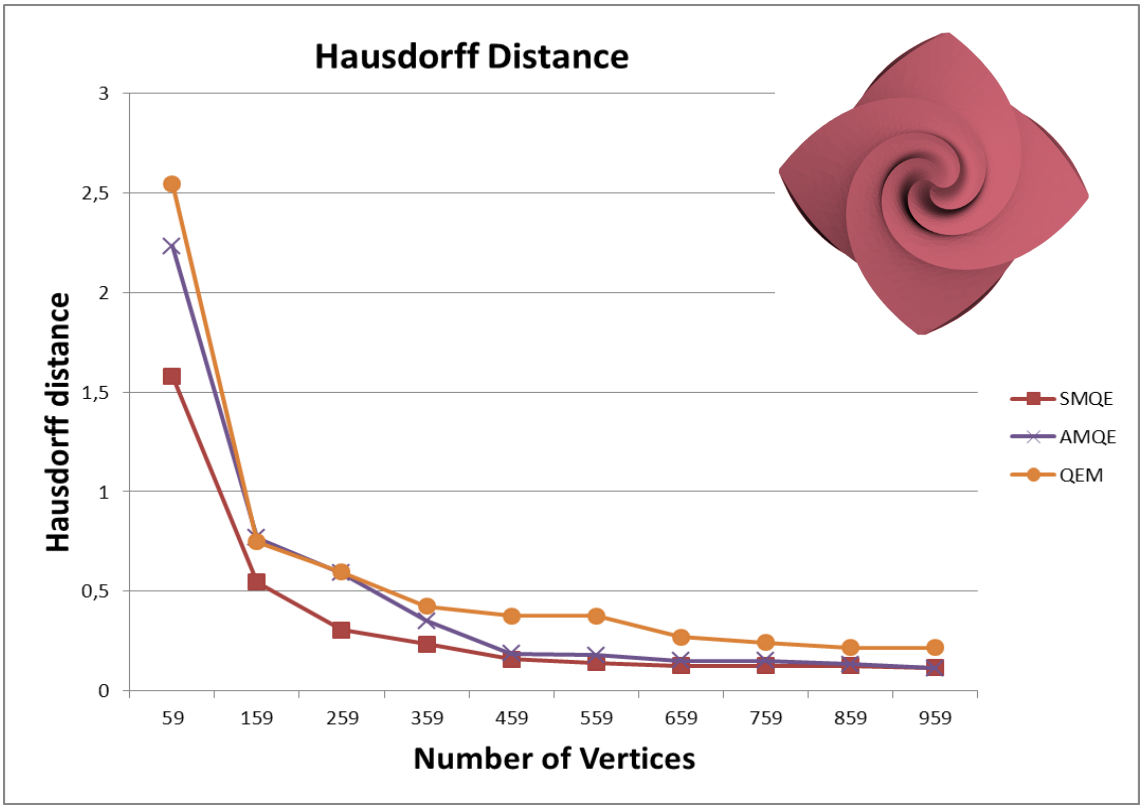


Figure 5.9: The Hausdorff distance introduced in the simplification of the Octa-flower. The Hausdorff distance is computed using the Metro tool.

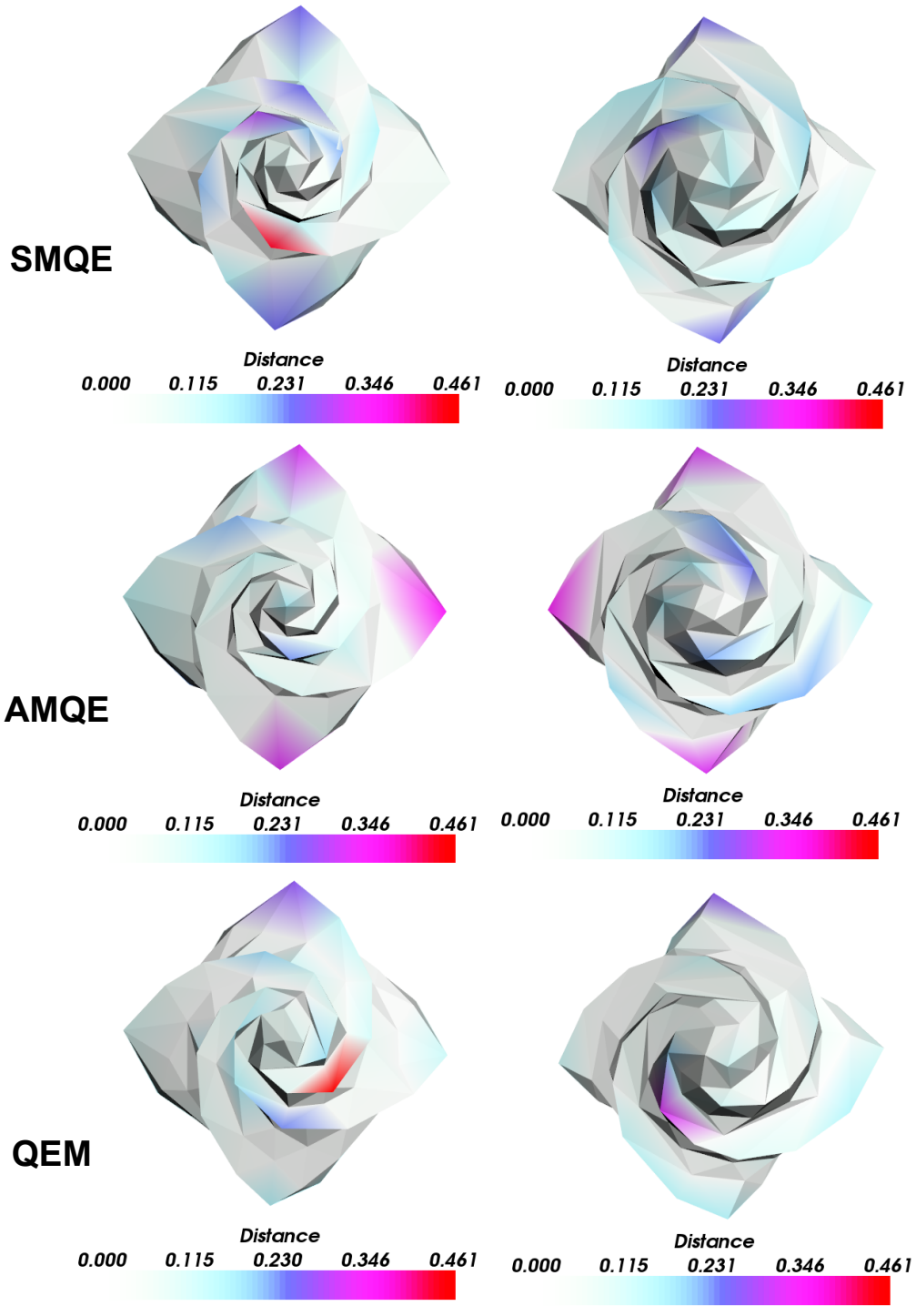


Figure 5.10: The distances between the Octa-flower with 7919 vertices and its simplified version with 169 vertices. The distances are computed between each vertex of the original model to the simplified model using the PQP library.

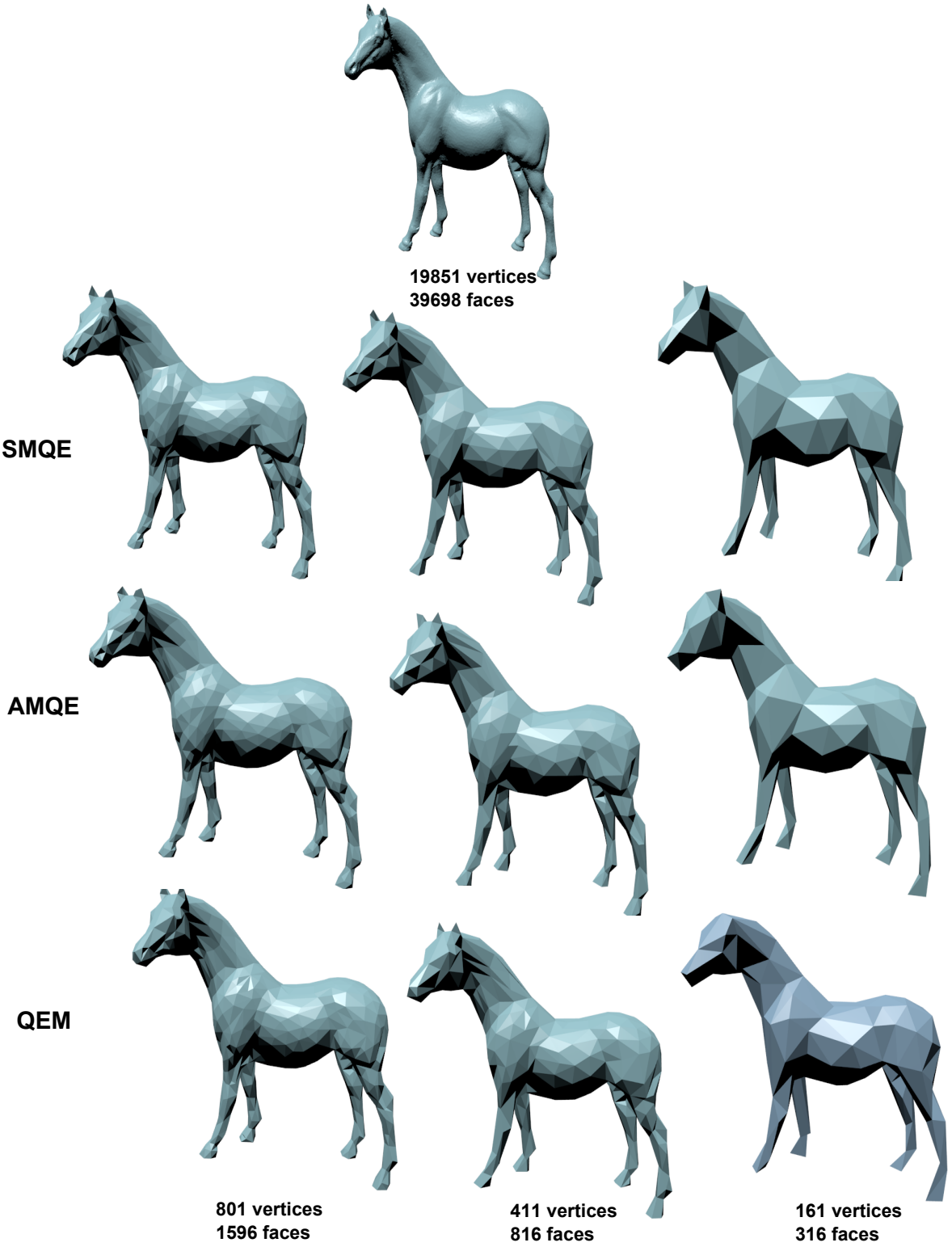


Figure 5.11: Sequence of simplifications for the Horse model (19 851 vertices) with 801, 411 and 161 vertices, respectively. The simplifications are realized using the SMEQ, AMQE and QEM methods.

On the other hand, if we examine the right front leg, its form seems better preserved for the models simplifed with AMQE and QEM than with SMQE. The quadratic error introduced by the simplification is smaller than for the model simplified with AMQE than the ones simplified with SMQE and QEM (quadratic error is $2.2907 \cdot 10^{-7}$, $2.3243 \cdot 10^{-7}$ and $2.7302 \cdot 10^{-7}$).
The Hausdorff distance between the simplified model and the original is smaller for the simplification with SMQE (0.003569). For the simplifications realized with the AMQE and QEM, the Hausdorff distance is quite similar (0.0077 for the model simplied with AMQE and 0.0080 for the one simplified with QEM).
The values for the quadratic error can be observed in the graph from the Figure 5.13. The quadratic error is computed for several simplified models, and for all of them, the error introduced by QEM is higher than for the models simplified with SMQE and AMQE.
On the other hand, the quadratic errors for the models simplified with SMQE and AMQE are almost identical.
The Hausdorff distance (Figure 5.14) is smaller for the the model obtained with SMQE. For the model with 411 and 681 vertices, the Hausdorff distance is higher than for the model with the same number of vertices but simplified with QEM.
The differences between the models simplified with SMQE and with AMQE are more visible for massive simplifications (model with lesser than 10% of the original number of vertices).
Based on this observation, in order to improve the running time of the simplification algorithm, we simplified the Horse model down to 2000 vertices using AMQE. Starting with this simplified model, we simplified using SMQE.

Dragon model:

For the Dragon model (1257 vertices), the features are better preserved for the model simplified with SMQE than for the one simplified with QEM (Figure 5.16). Features such as the tongue of the Dragon or the first left foot are closer to the original for the simplification with SMQE than for the one with QEM.
In Figure 5.17, the curve for the quadratic error obtained with the SMQE is below that obtained with QEM.
The Hausdorff distances are shown in Figure 5.18. The values of the Hausdorff distances are almost equal for the models obtained using SMQE and AMQE and higher for the models obtained using QEM.
The Hausdorff distance is $0,003638$ for the models obtained using SMQE and AMQE and $0,008381$ for the one obtained using QEM. All the simplified models have 707 vertices. The Quadratic errors for these models are 0.1353314, 0.1355278 and 0.1795107 respectively.

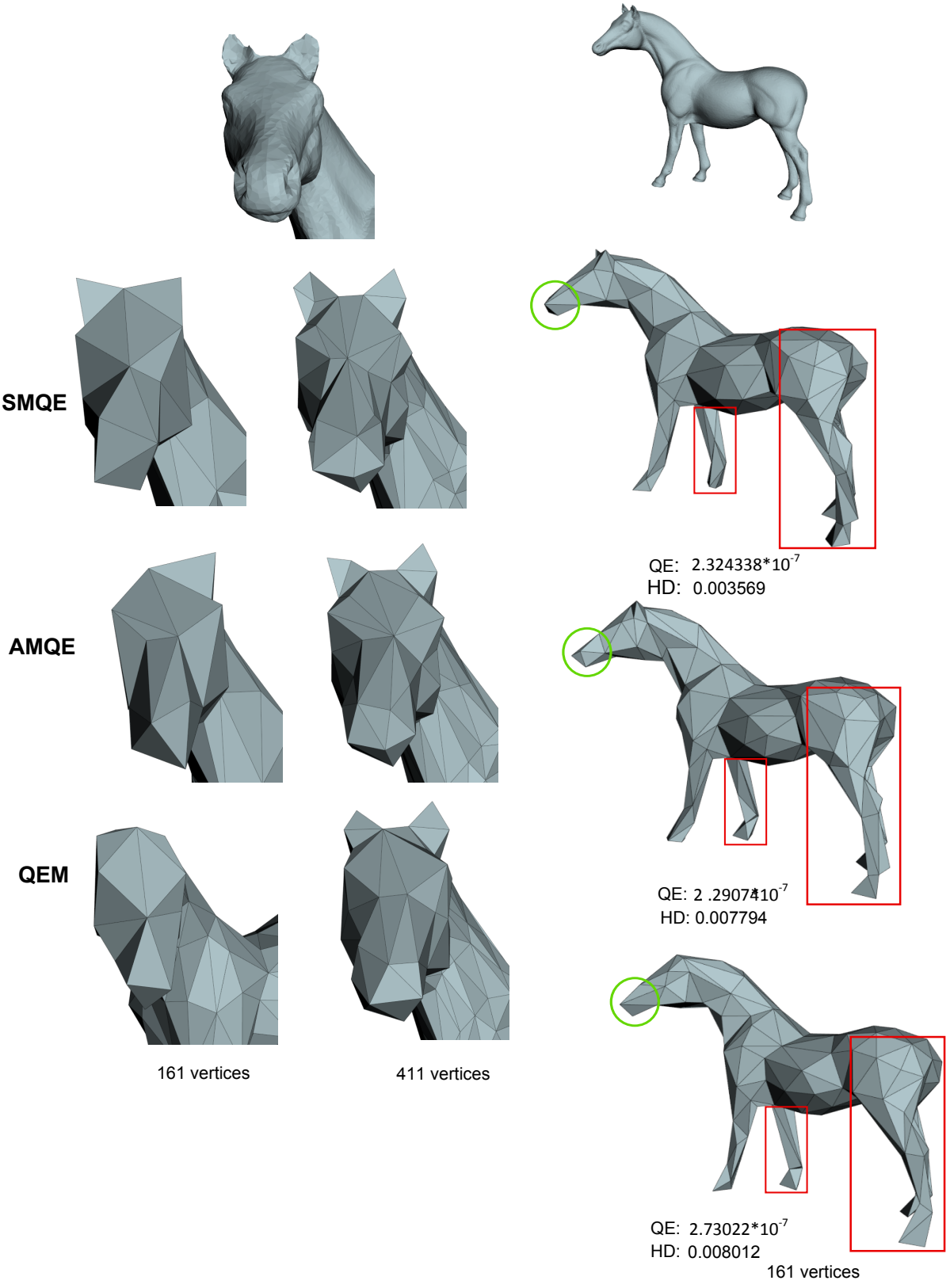


Figure 5.12: Details for simplified Horse with 161 and 411 vertices.

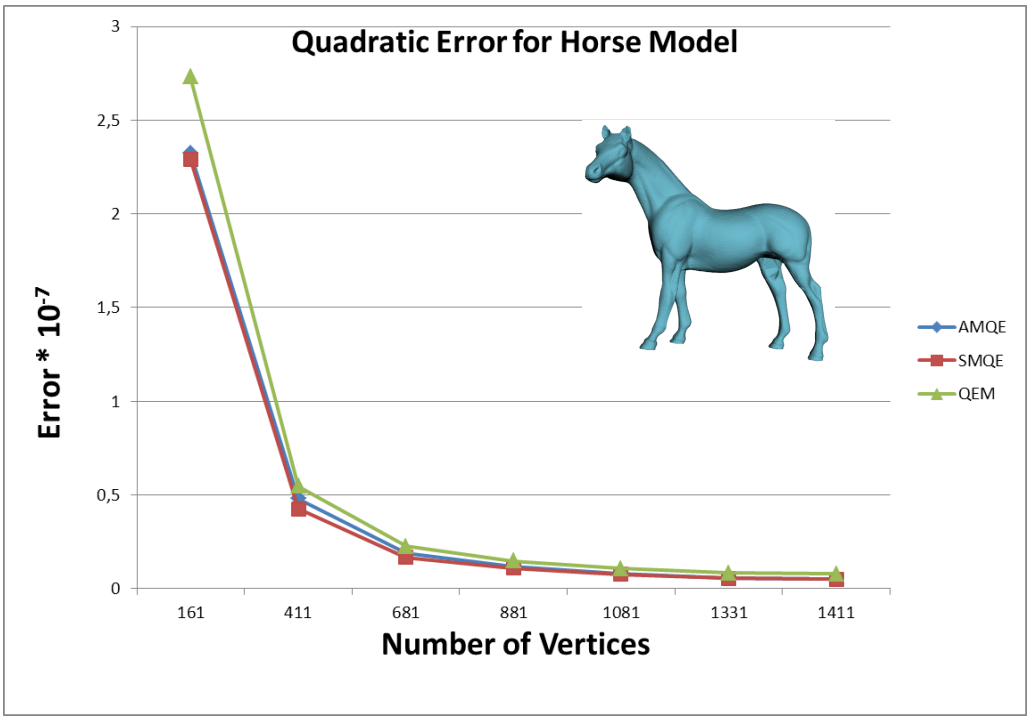


Figure 5.13: The quadratic error introduced in the simplification of Horse.

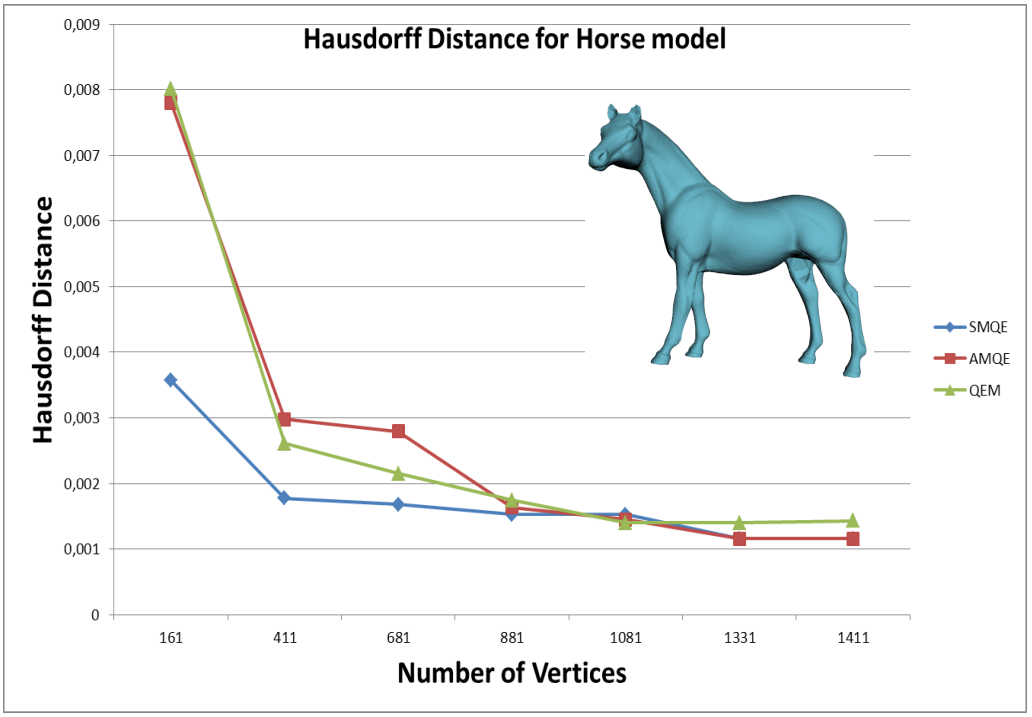


Figure 5.14: The Hausdorff distances introduced in the simplification of Horse. The Hausdorff distances are measured using the Metrol tool.

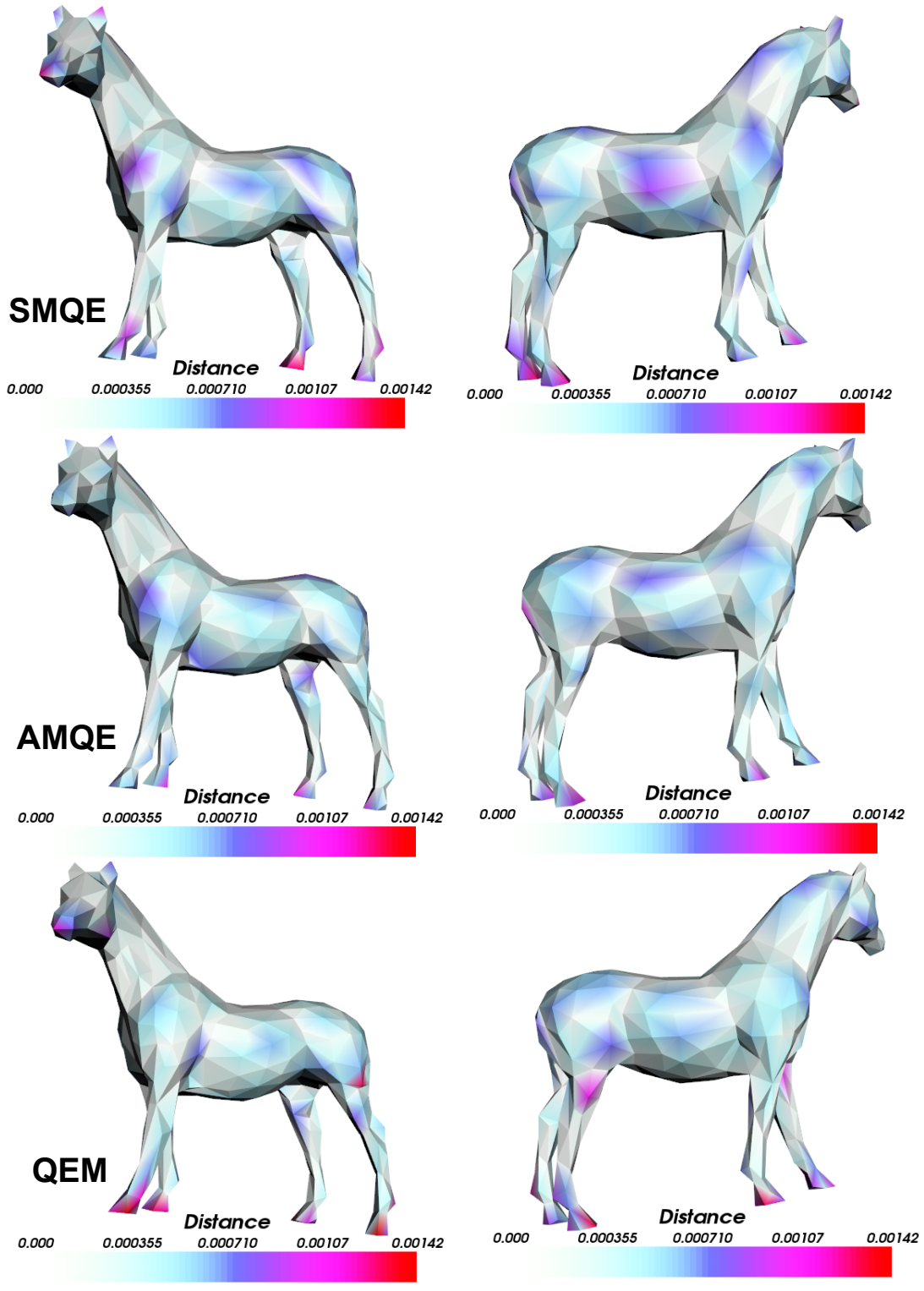


Figure 5.15: Distances between the Horse model with 19 851 vertices and its simplified version with 411 vertices. The distances are computed between each vertex of the original model to the simplified model using the PQP library.

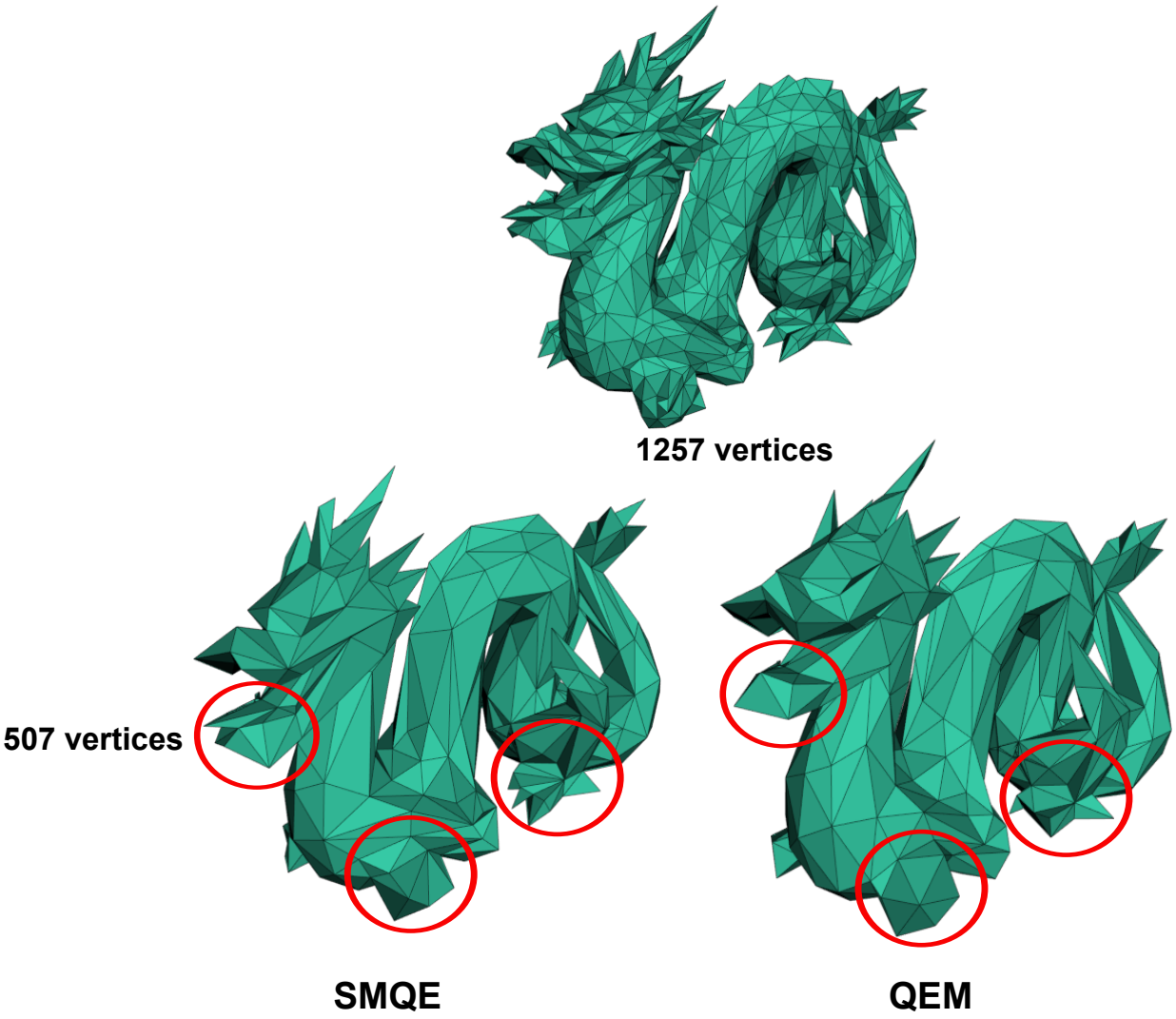


Figure 5.16: The simplified versions of the Dragon (1257 vertices) with 507 vertices using the SMQE and QEM methods.

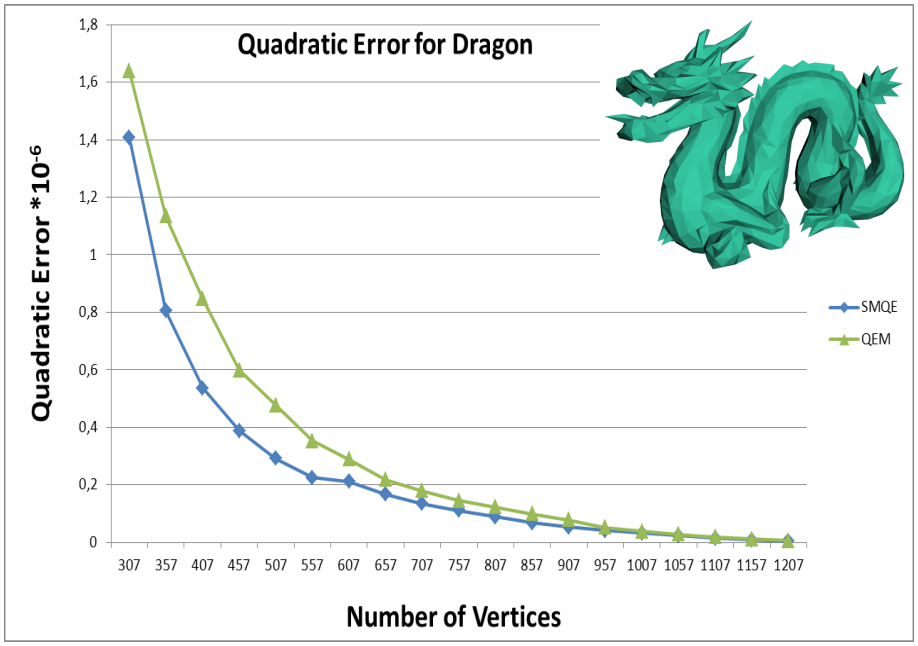


Figure 5.17: The quadratic errors introduced in the simplification of the Dragon.

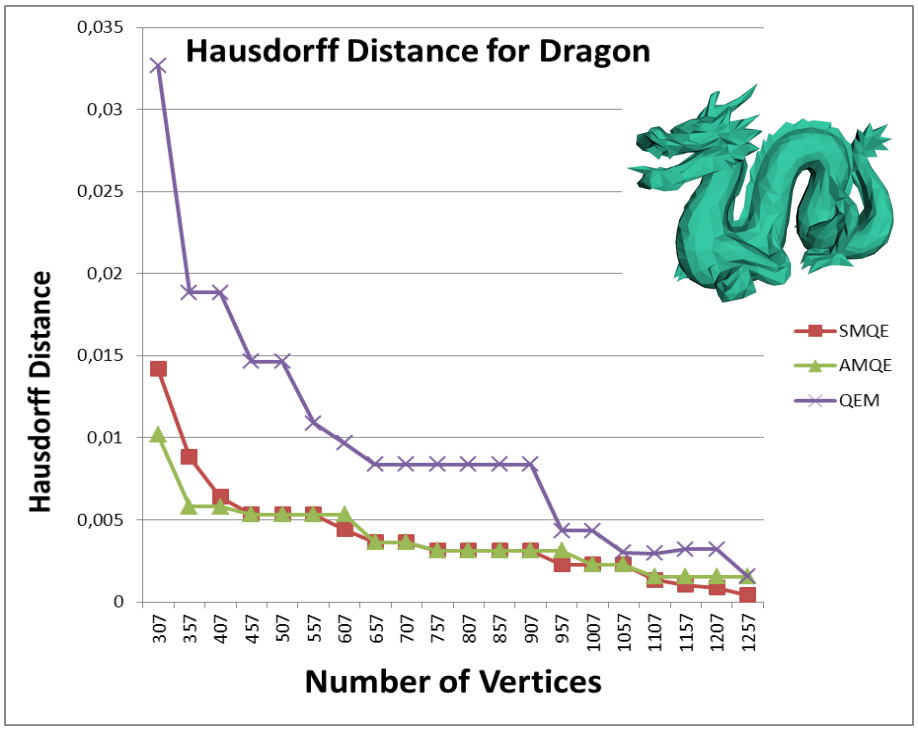


Figure 5.18: The Huasdorff errors introduced in the simplification of the Dragon.

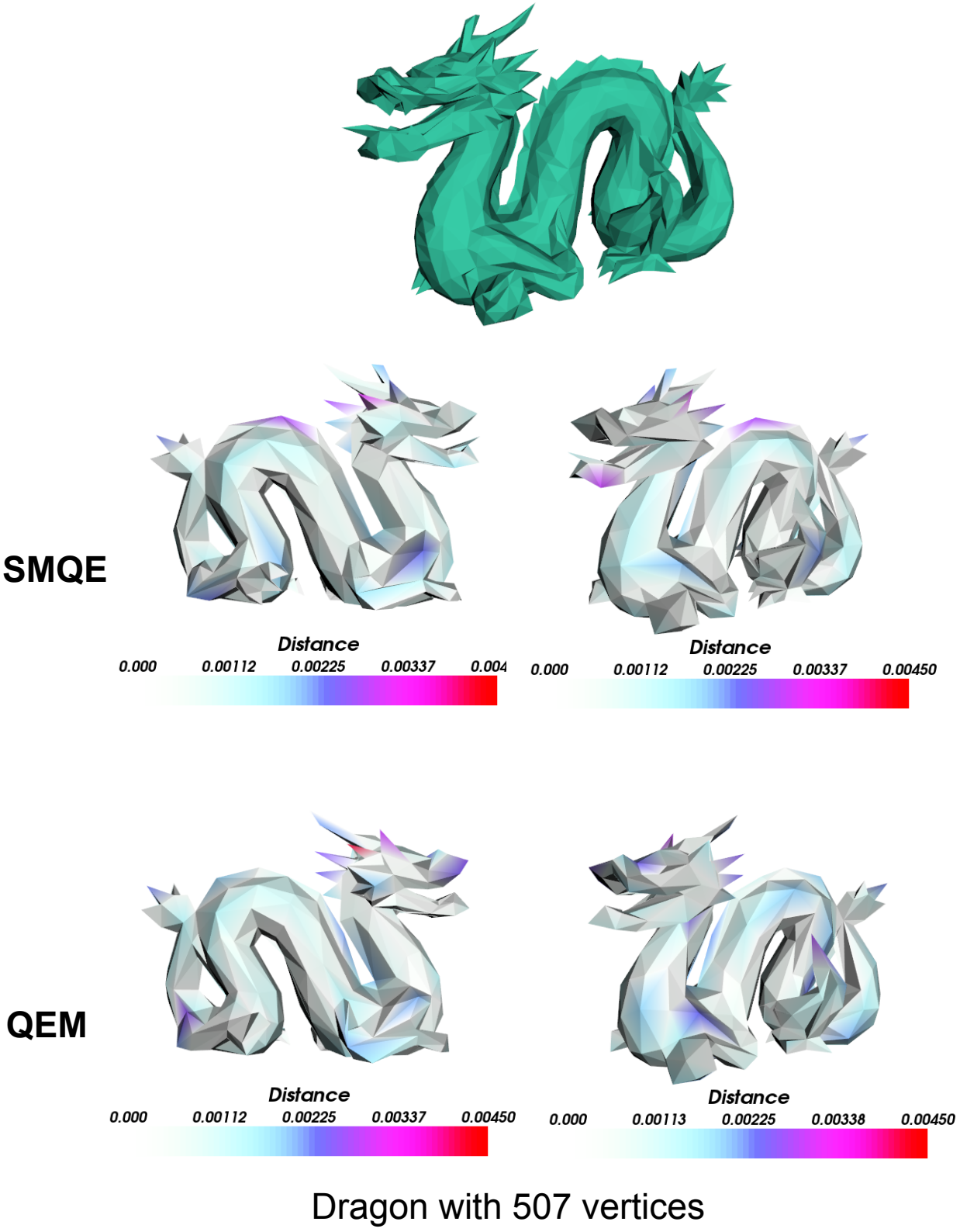


Figure 5.19: The distances between Dragon (1257 vertices) and its simplified version with 507 vertices using the PQP library.

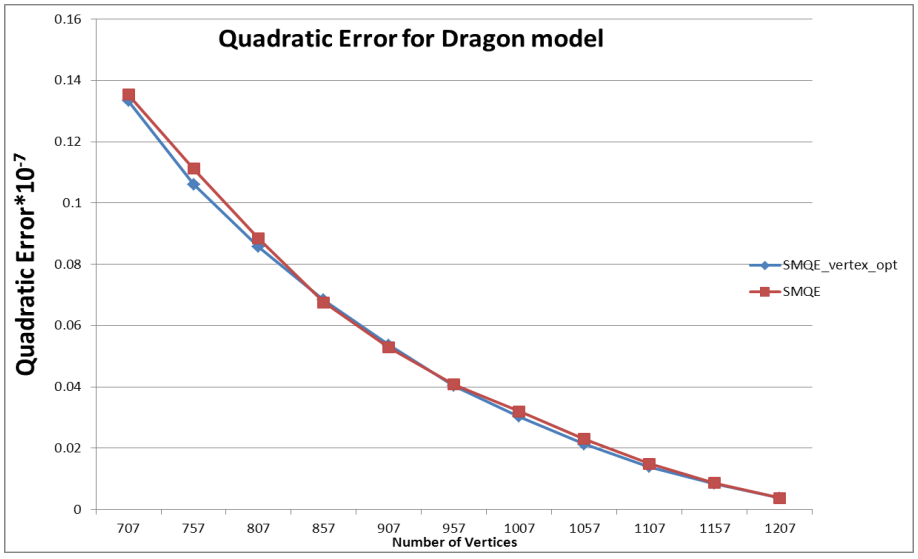


Figure 5.20: The distances between Dragon (1257 vertices) and its simplified version with 507 vertices using PQP library.

5.2 Boundary Preservation

Bunny model:

An important advantage of using SMQE is the preservation of the islands and boundaries of a mesh. For example, the Bunny model (35 947 vertices) has 4 holes in its base (Figure 5.21). When the model is simplified, the boundaries (red lines) are preserved for the models simplified with SMQE while those simplified with AMQE are not so accurately preserved.

Beethoven model:

In Figure 5.23, the Beethoven model is simplified until a model with 505 vertices using both SMQE and AMQE error metrics. Form Figure 5.23 we can see the boundaries around the eyes and around the face are better preserved for the model simplified with SMQE than for the one simplified using AMQE. On the model simplified with AMQE, the faces almost disappear.

Moreso, the quadratic errors introduced into simplification by the SMQE are lesser than the quadratic errors introduced by AMQE which in turn are lesser than those introduced by QEM (see Figure 5.24).

The hair waves and the details on Beethoven's bow are better preserved for the model simplified with AMQE. The triangles simplified on the face for the model obtained with AMQE are used to represent the hair and the bow.

The Hausdorff distance introduced by SMQE is lower than the one introduced by AMQE (see Figure 5.25). For example, for the Beethoven model with 405 vertices simplified with SMQE, the Hausdorff distance is 0.24415 while for the Beethoven model simplified with AMQE it is 1.510782. For the Beethoven model with 1405 vertices simplified with SMQE,

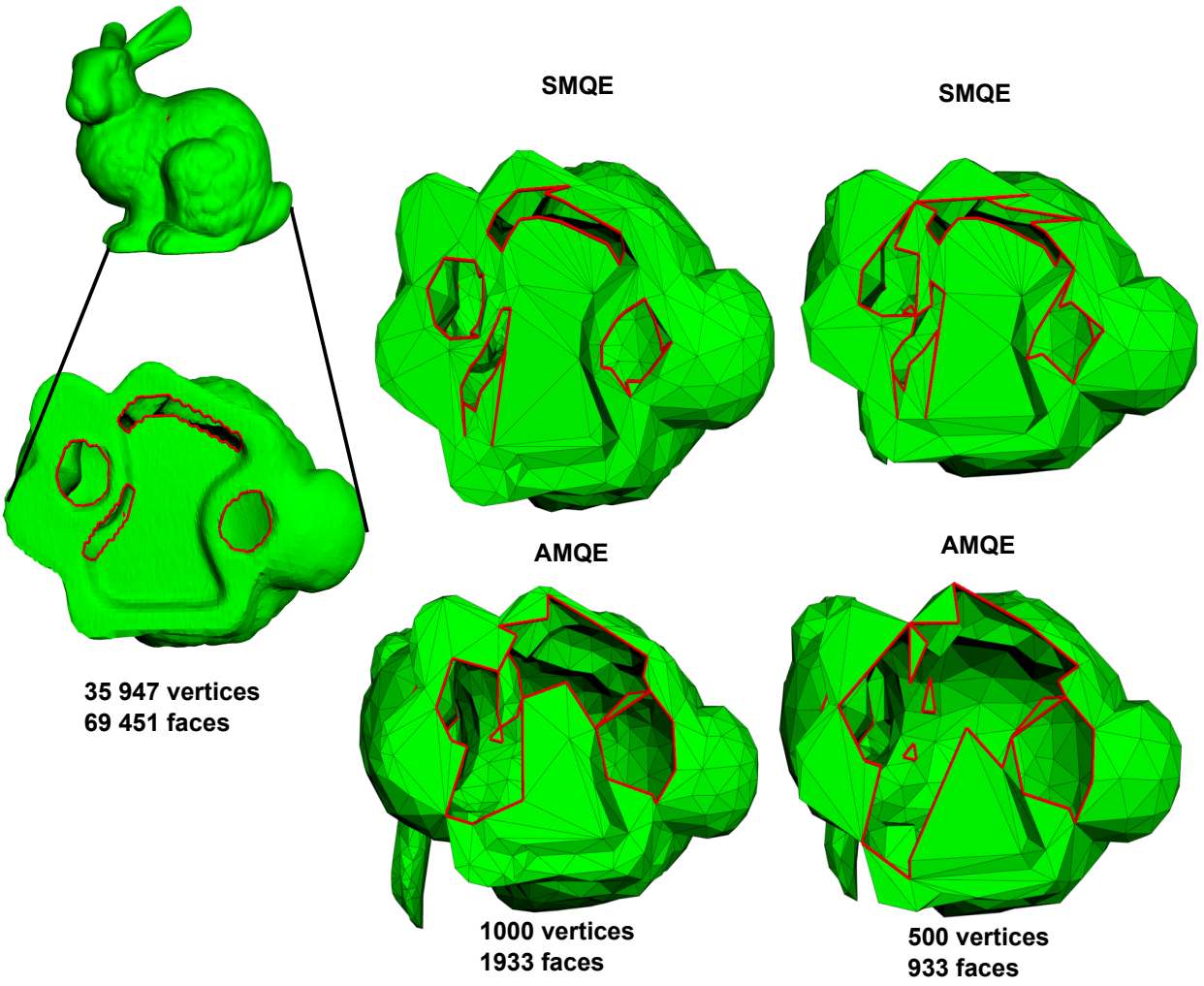


Figure 5.21: The base for the Bunny model (first column) and its simplified versions with 1000 vertices, respectively 500 vertices. The boundaries (red) are better preserved for the model simplified with SMQE (top) than for the model simplified with AMQE (bottom).

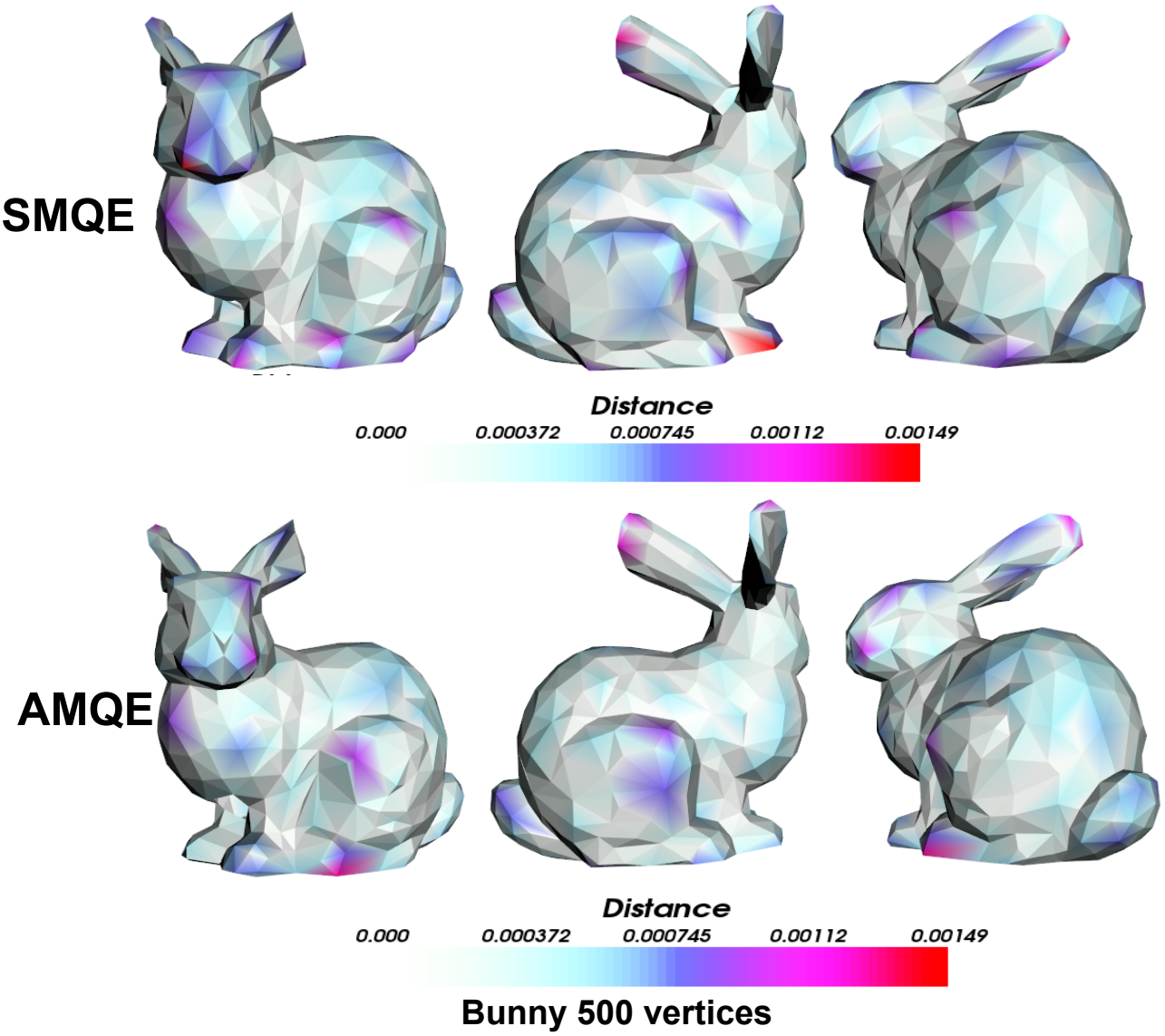


Figure 5.22: The distances between the original Bunny (35 947 vertices) and its simplified version with 500 vertices. The distances are computed between each vertex of the original model to the simplified model by using the PQP library.

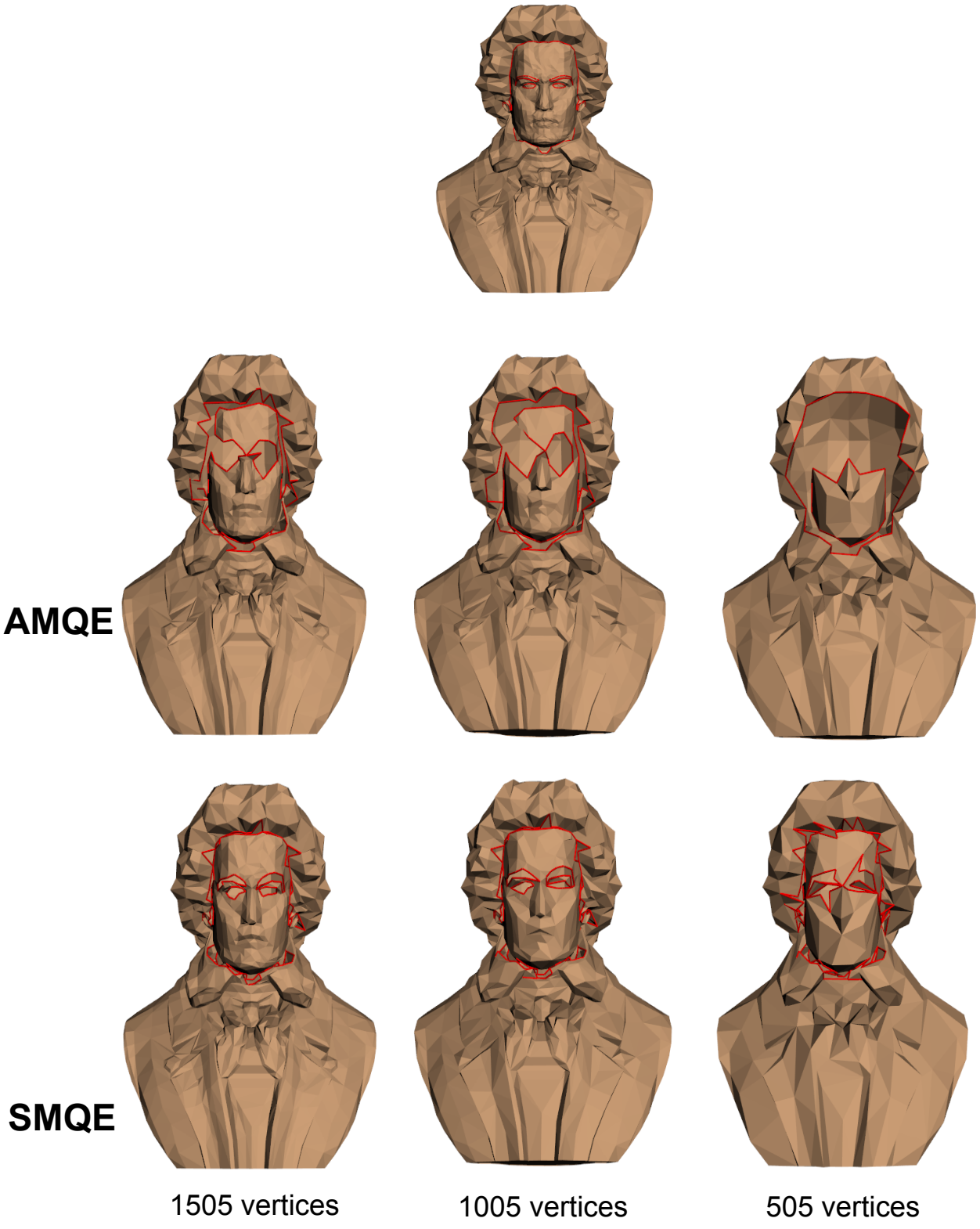


Figure 5.23: A sequence of simplifications for the Beethoven (2655 vertices) model with 1505, 1005 and 505 vertices. The boundaries (red lines) are better preserved for the simplifications realized with the SMQE method than with AMQE.

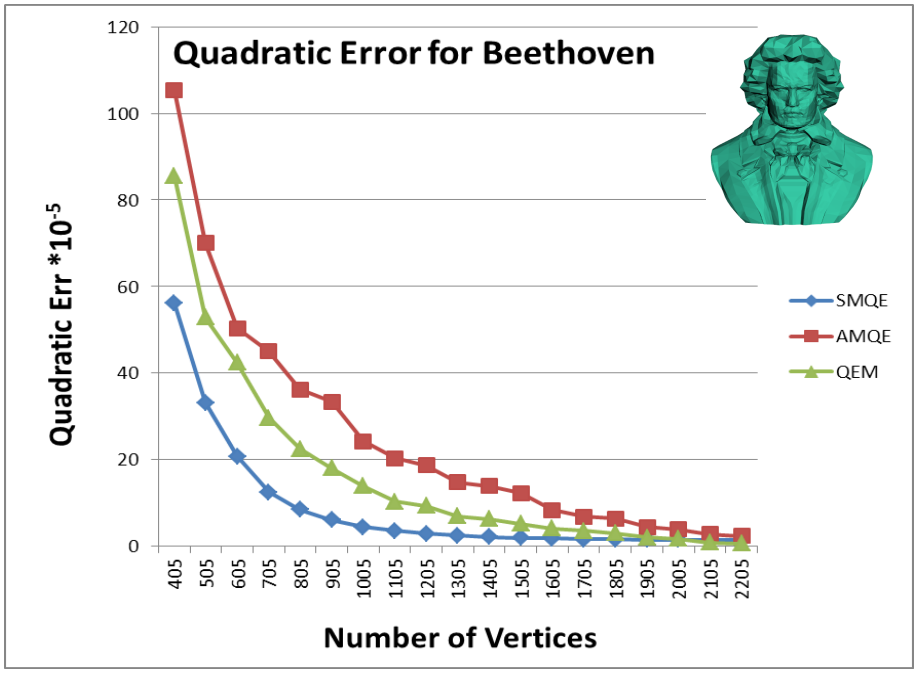


Figure 5.24: The quadratic errors introduced in the simplification of Beethoven.

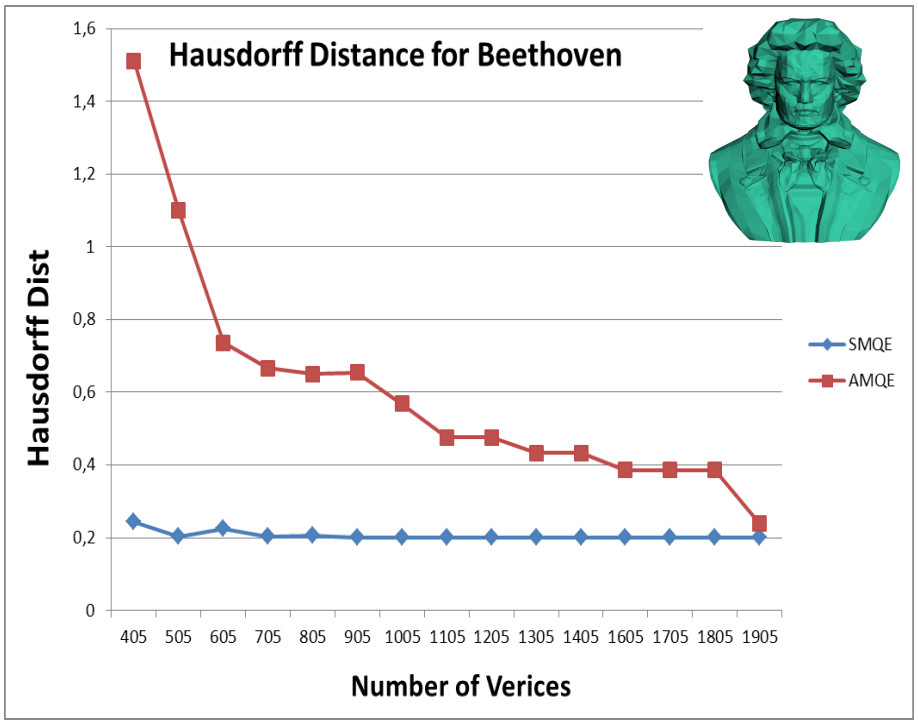


Figure 5.25: The Hausdorff distances introduced in the simplification of Beethoven measuread with the Metro tool.

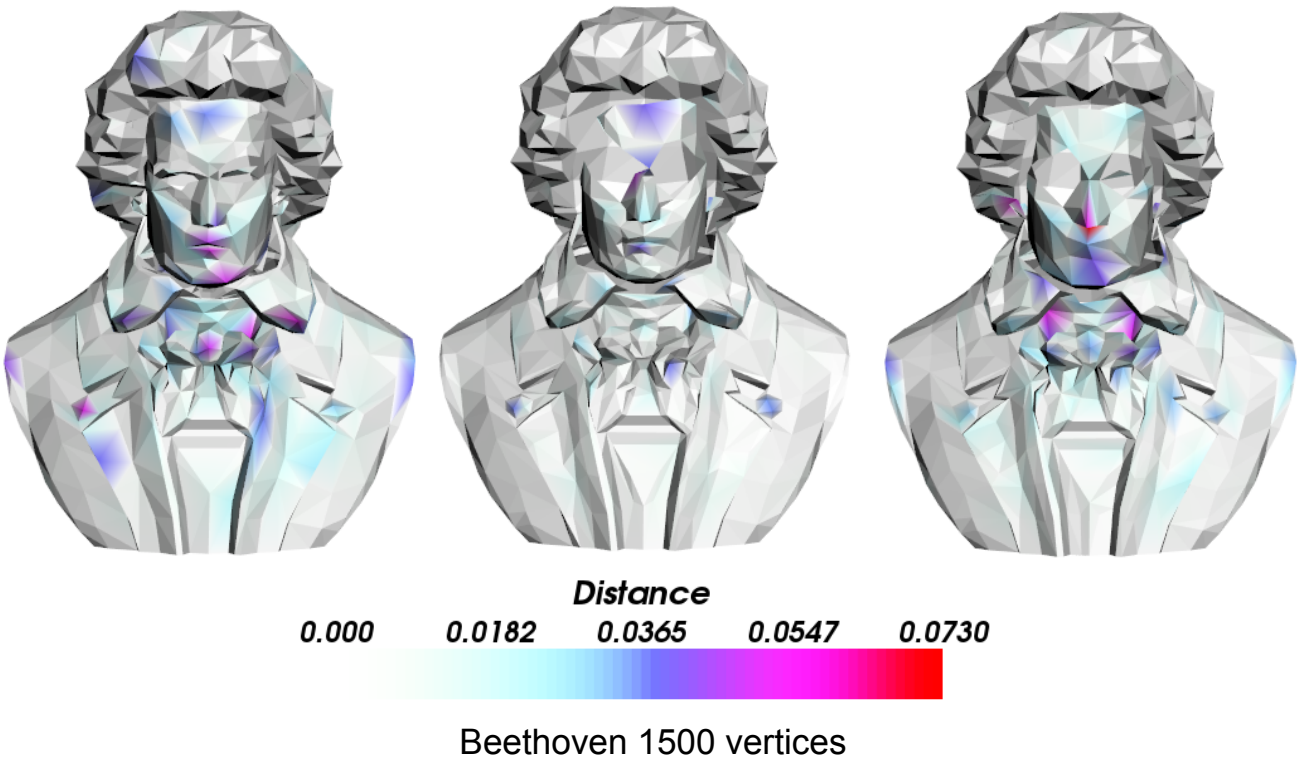


Figure 5.26: The distances between the original Beethoven (2655 vertices) and its simplified version with 1500 vertices. The distances are computed between each vertex of the original model to the simplified model by using the PQP library.

the Hausdorff distance is 0.200734 and 0.432407 for the model with the same complexity, but simplified with AMQE.

Venus model:
In Figure 5.27, the Venus model (8628 vertices) is simplified until 500 vertices using SMQE and QEM error metrics. From the figure, we can see the details (such as the nose) are better preserved for the model simplified using SMQE error metric. Details such as the lips and the waves of the hair are better outlined. The quadratic error for the model simplified with SMQE is 0.1189699 and the Hausdorff distance is 0.007549 while, for the one simplified with QEM is 0.011258 and the quadratic error is 0.1561323.
For the forehead of the Venus model, which is almost flat, the QEM uses more triangles than the SMQE. Thus, more triangles are used by the SMQE to approximate curved regions such as the loop of hair of the Venus model or the waves of the hair (Figure 5.28).
As expected, the quadratic error introduced into the simplfications by SMQE is below the quadratic error introduced by QEM (Figure 5.29). The same stands for the Hausdorff distance (Figure 5.30).
We do not present the results for the simplifications obtained with AMQE because they are very close to those obtained with SMQE.
Bones model:

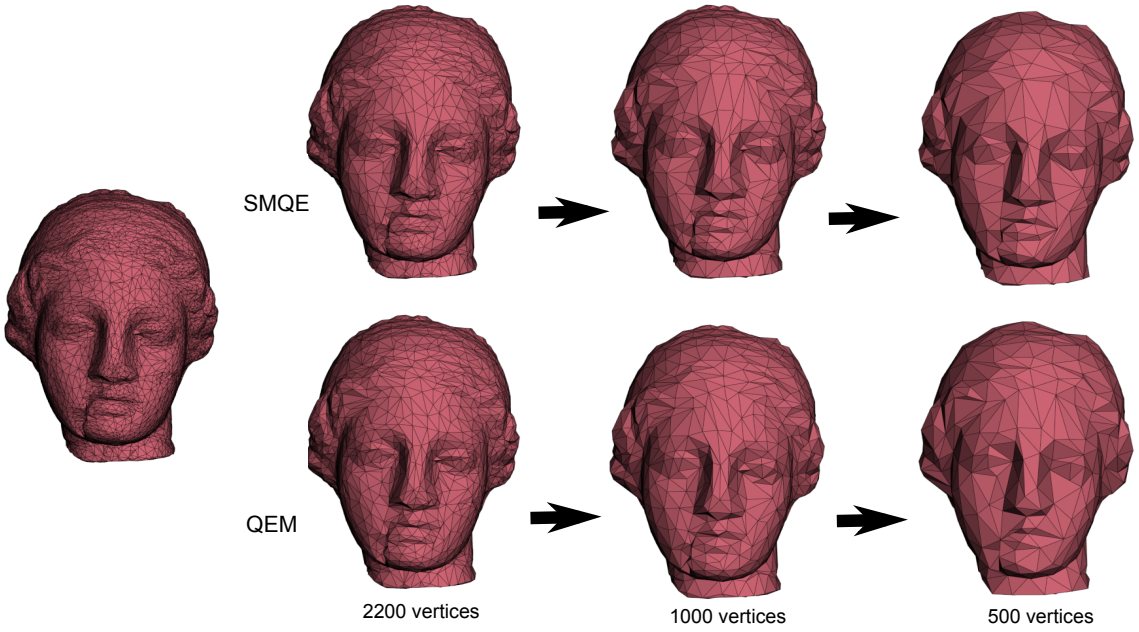


Figure 5.27: The sequence of the simplifications for the Venus model (8268) with 2200, 1000 and 500 vertices using the SMQE and QEM methods.

Figure 5.32 shows the Bones model (2154 vertices) together with its simplified versions with 444 vertices obtained with both SMQE and QEM. For early simplifications the quadratic error metric measured for the simplifications obtained with SMQE (Figure 5.33) is lesser than the quadratic error measured of simplifications obtained with QEM. During simplification, QEM eliminates some details such as the triangles of the distal phalanges and uses these triangles for other regions of the model while SMQE preserves them.
For this reason, the Quadratic Error is bigger for massive simplifications, such as for the model with 594 vertices simplified using SMQE error metric (see Figure 5.33, the bottom part) than for the same number of vertices model but simplified using QEM error metric. We notice that the triangles of the medial and distal phalanges almost disappear for the model simplified with QEM. This is not the case for the model simplified with the SMQE, where the triangles are not simplified.
In the Bones model's simplification, we replace the resulting vertex using Volume-based vertex optimization (see Section 4.6).
Fandisk model:
In Figure 5.36, the Fandisk model is simplified using SMQE error metric. We can remark from this figure, the SMQE preserves the boundaries even for a model created using CAD (Computer Aided Design) techniques.
In conclusion, the presented results are obtained using an accurate measure (AMQE) and a symmetric measure of the quadratic error introduced by an edge collapse. The resulting vertex (the vertex obtained from an edge collapse) is placed in order to minimize the quadratic error from the QEM. The quality of our results is evaluated using the quadratic

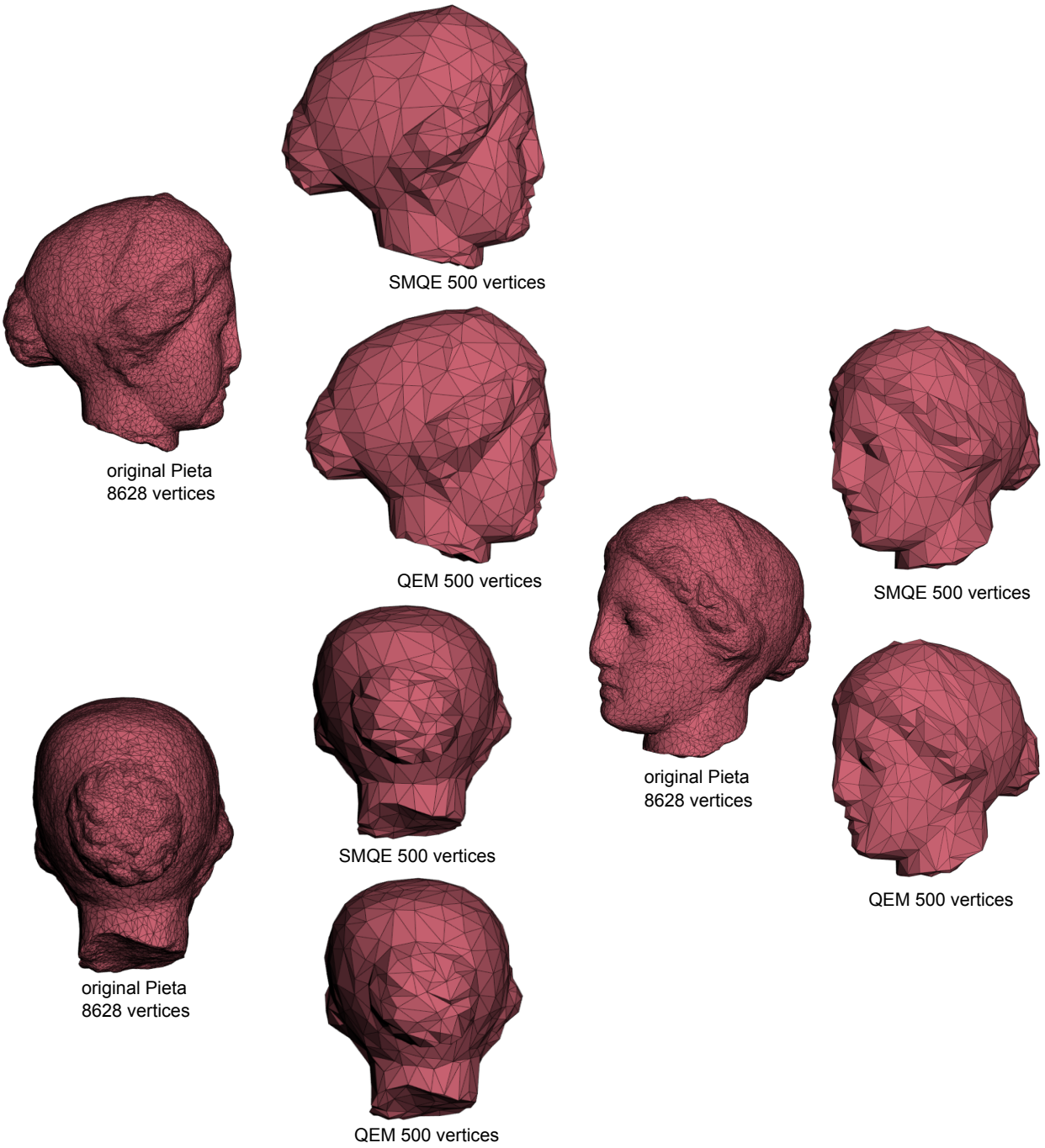


Figure 5.28: Simplified versions for the Venus model; multiple viewpoints.

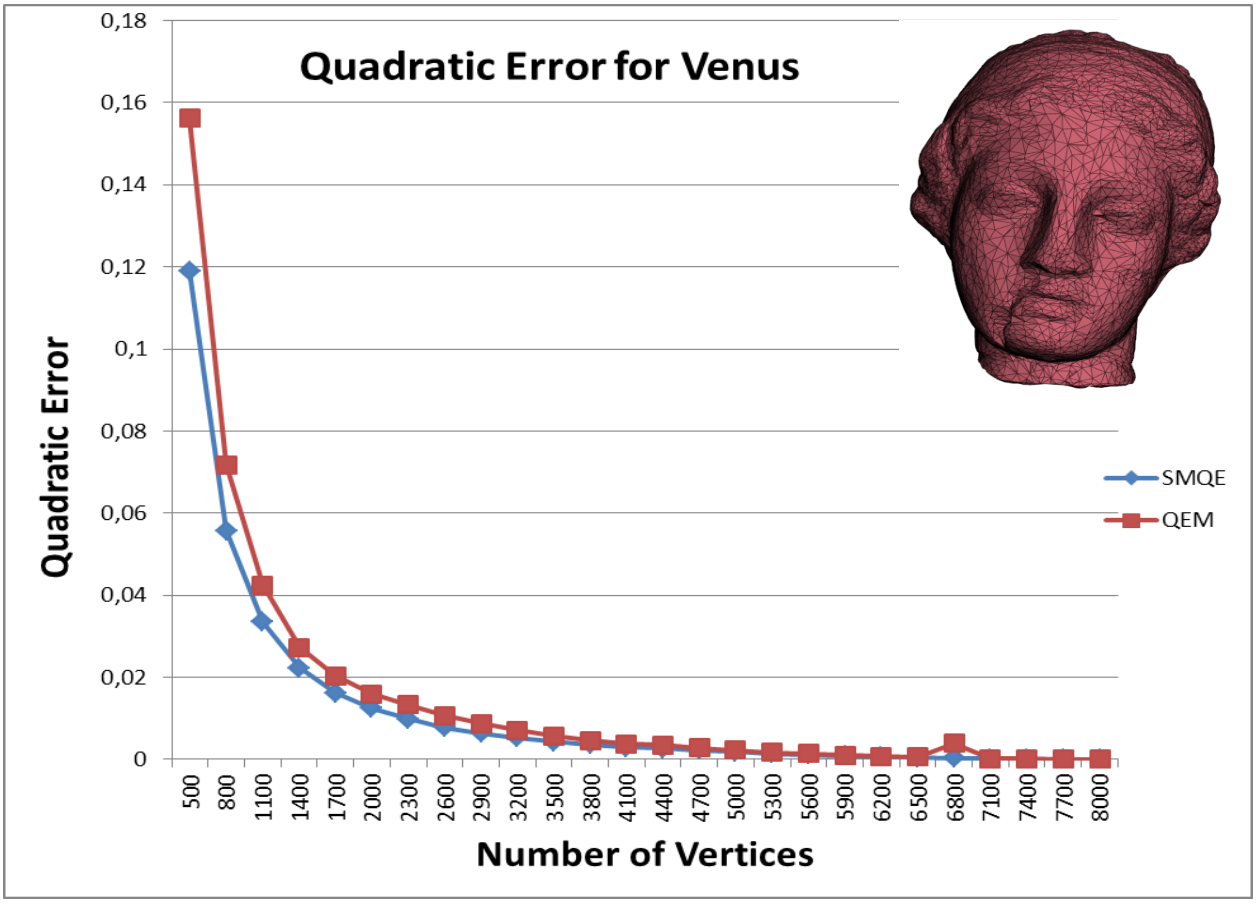


Figure 5.29: The quadratic errors introduced in the simplification of the Venus model.

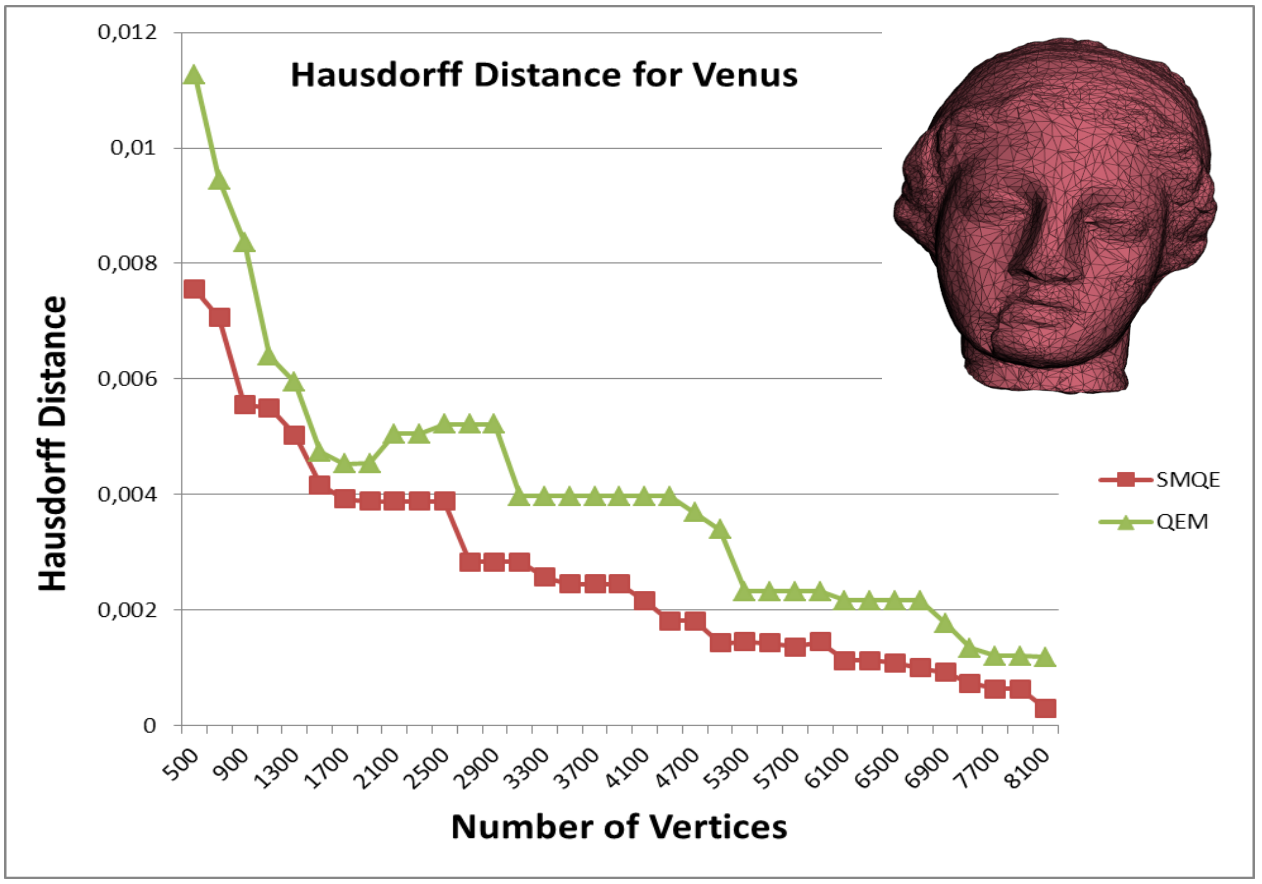


Figure 5.30: The Hausdorff distances introduced in the simplification of the Venus model, measured using the Metro tool.

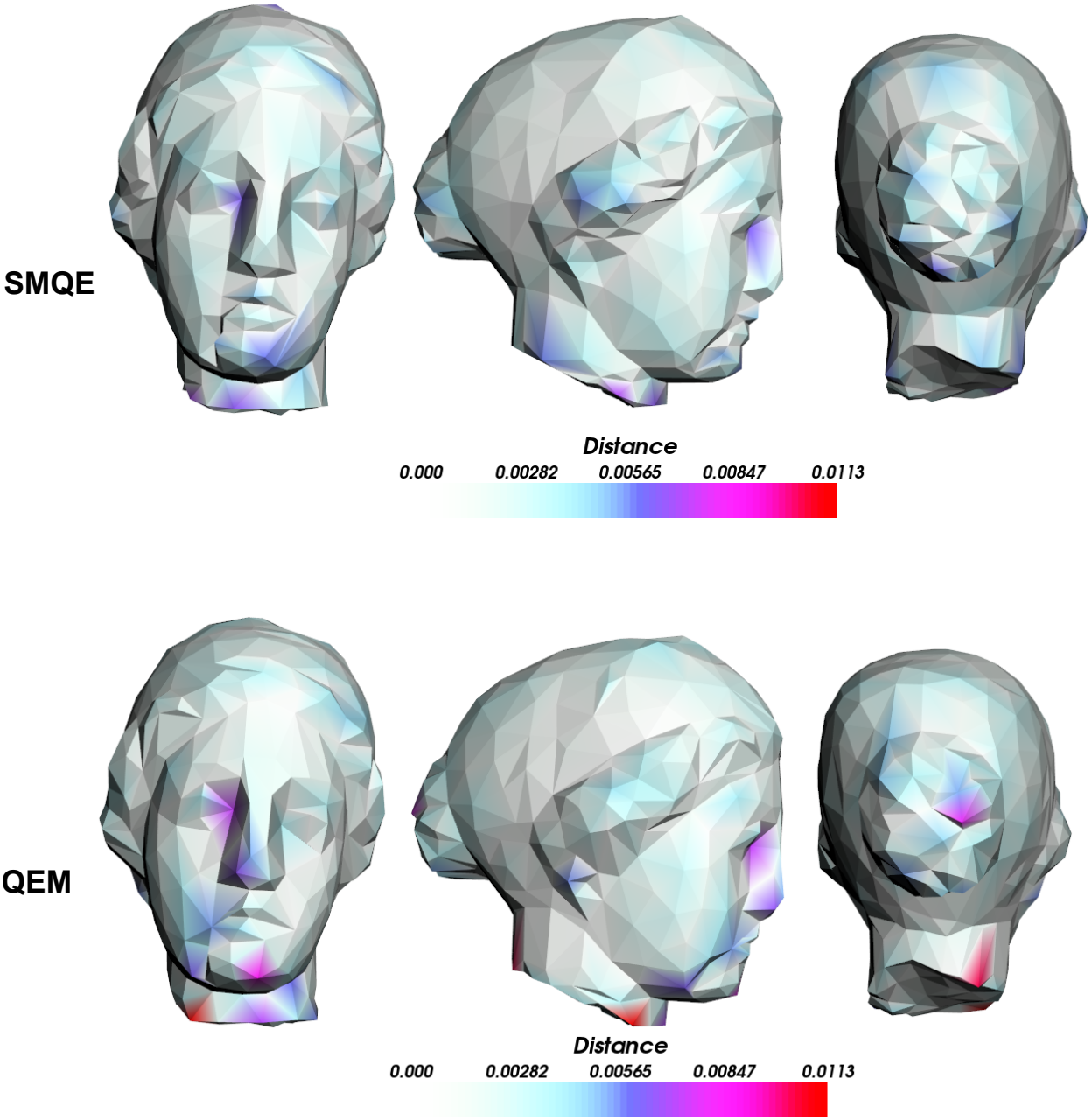


Figure 5.31: The distances between the Venus model and its simplified version with 500 vertices.

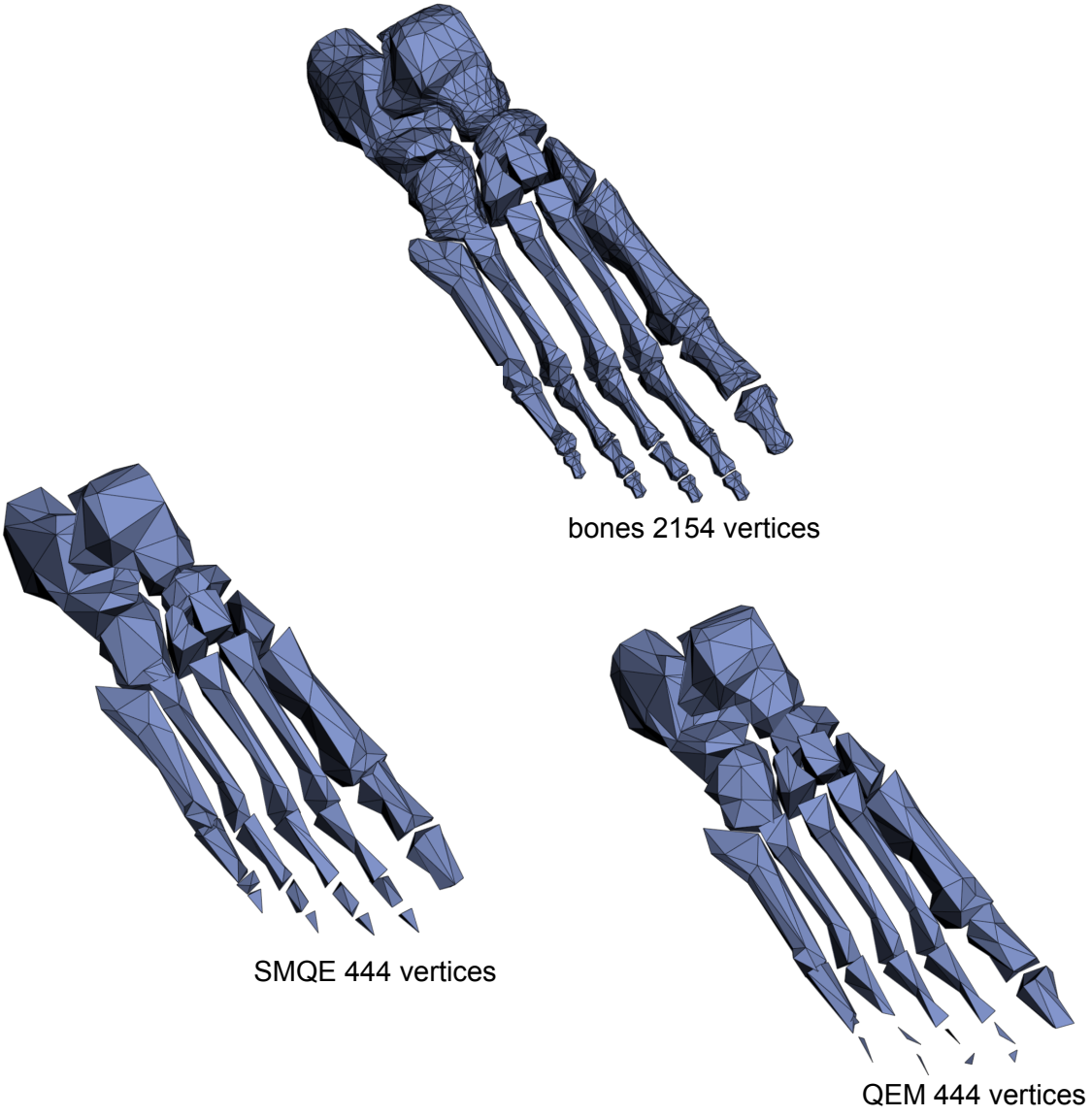


Figure 5.32: Simplifications for the Bones model.

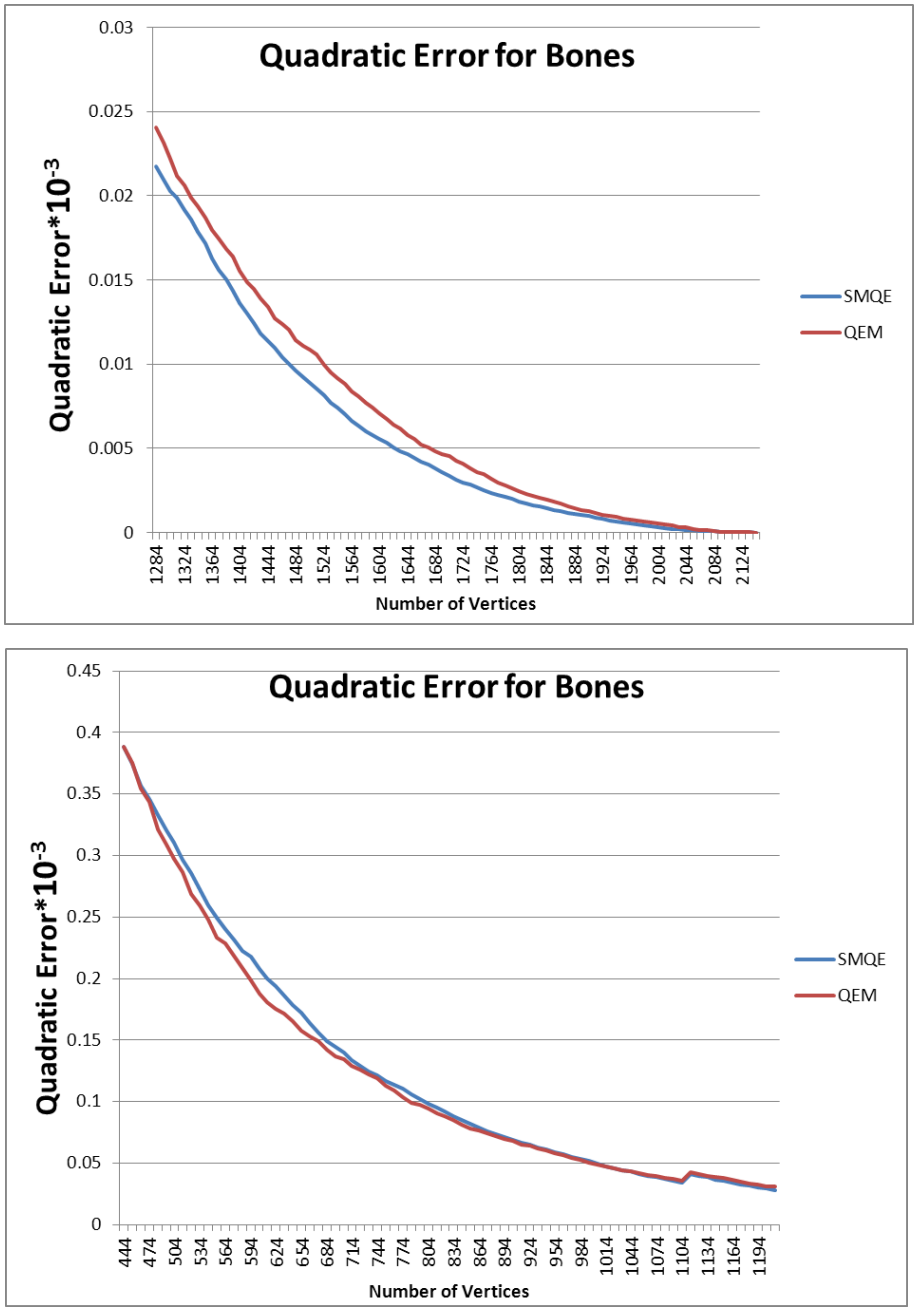


Figure 5.33: Quadratic Error for the Bones model vs. Number of Vertices

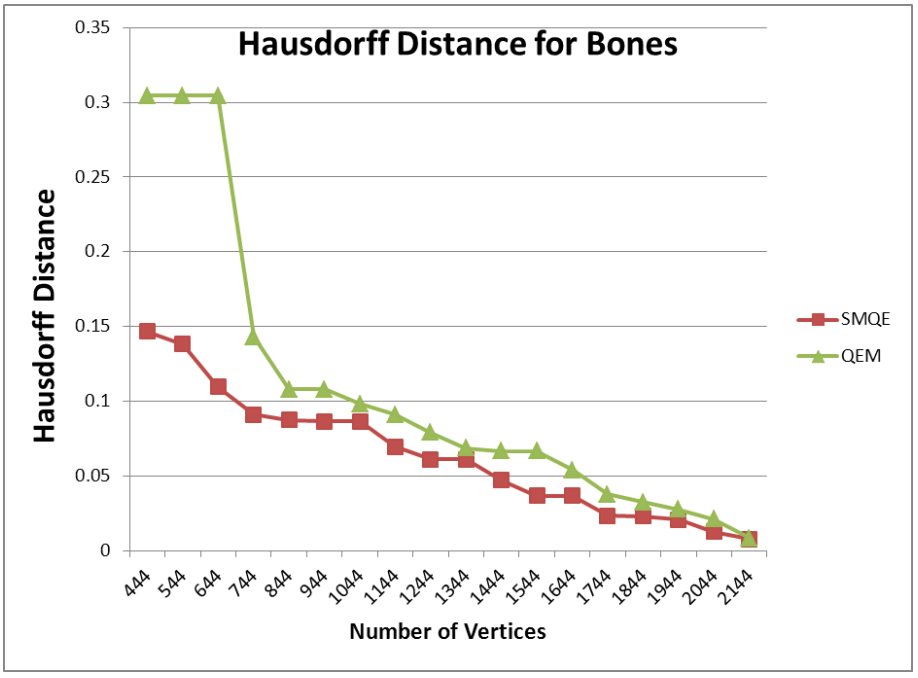


Figure 5.34: Hausdorff Distance for the Bones model vs. Number of Vertices. The Hausdorff distance is computed using the Metro tool.

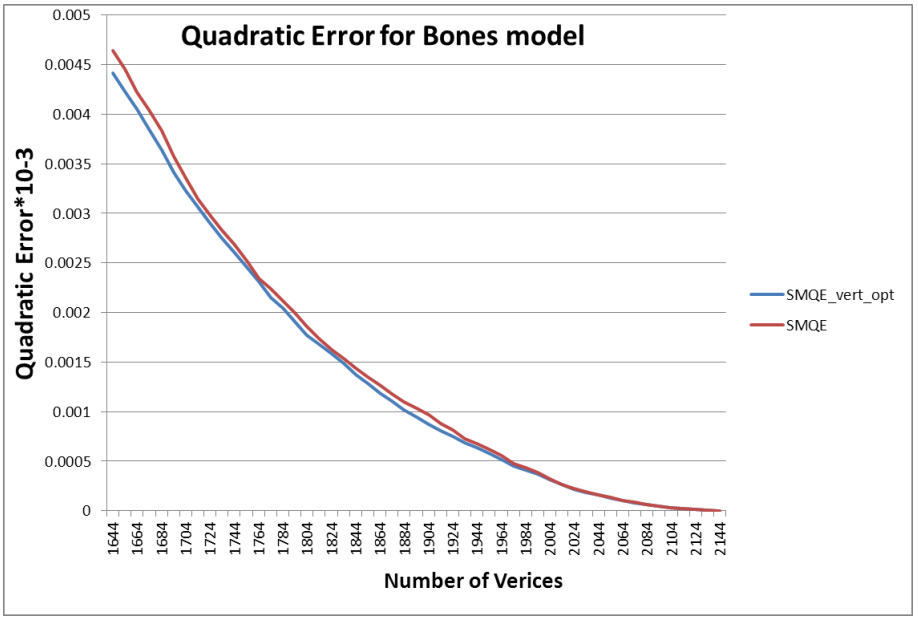


Figure 5.35: Quadratic Error for Bones model vs. Number of Vertices. $SMQE_{vert-opt}$ represents the error for the model simplified with SMQE and using Volume-based vertex optimization to locate the resulting vertex.

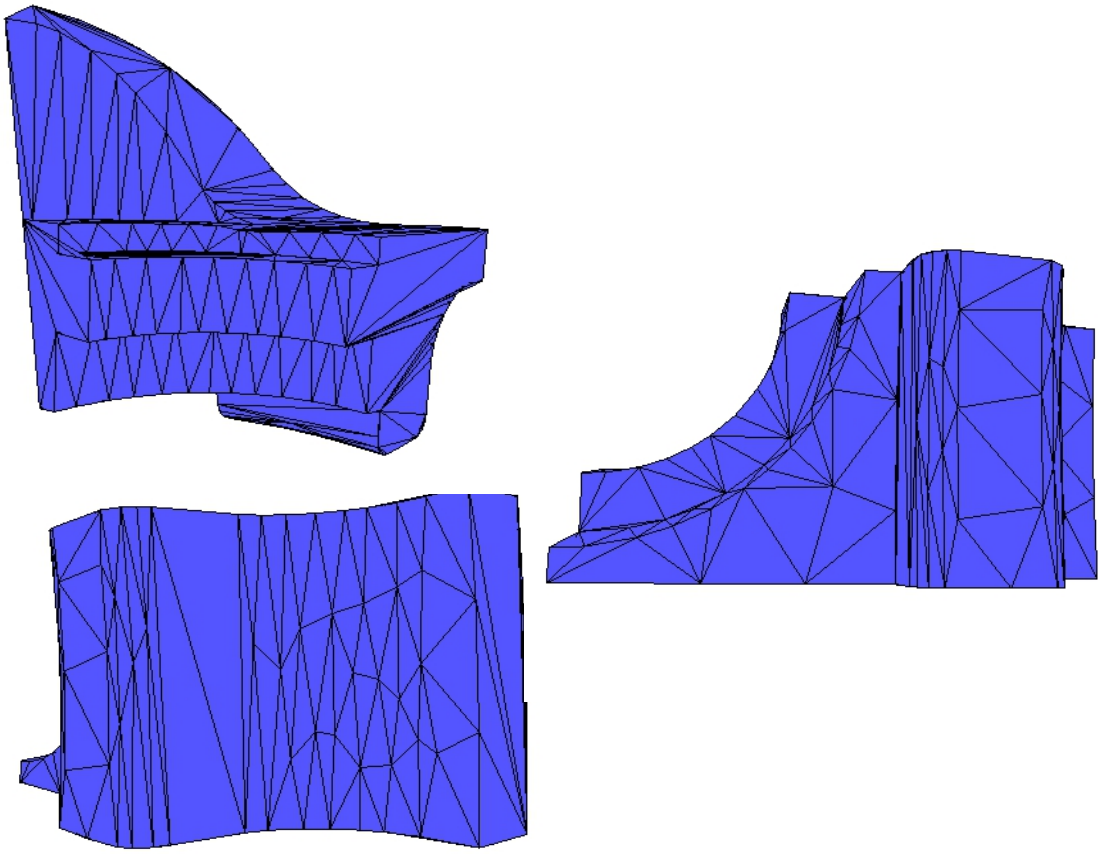


Figure 5.36: The simplifications of Fandisk model obtained using SMQE error metric (displayed from multiple points of view).

error and the Hausdorff distance between the simplified models at different level of details and the original model.
A drawback of our algorithm caused by the high level accuracy is the time complexity. The running time can reach up to one day for a model with 50000 vertices. The time complexity is not so critical because our simplifications are made offline but characterized by high level of accuracy.

Conclusions and perspectives

We have presented an algorithm which produces accurate approximations for triangular meshes. The accuracy is given by the two geometric error measures used: AMQE and SMQE. The results obtained with SMQE are better than those obtained with AMQE because SMQE computes the distances between the meshes in a symmetric manner.
We compare our method with QEM, one of the references in literature. Our measurements provide similar results compared to QEM for the early stages of simplifications, whereas for massive simplifications (i.e. more than 90%) the quality of the results demonstrates the advantanges of using an accurate and symmetric measure.
For meshes with boundaries, the advantage of using a symmetric measure is proven by the boundary preservation during simplification. Compared to QEM, AMQE better preserves the boundary edges because of the accuracy of the distance computation.
Our algorithm reduces the geometry of a mesh preserving the topology. For certain applications such as medical vizualisation, preserving the topology is desirable. While, for topology changes caused by digitization artefacts, topology reduction is desirable.
To reduce topology complexity, the edge collapse operator can be replaced by pair vertices contraction. The complexity of pair vertices contraction is $n \cdot (n-1)$ where n is the number of vertices.
Because our error measures are computationally expensive, pair vertices contraction used as a simplification operator would increase the complexity of the algorithm.
In conclusion, topology simplification can be necessary but is expensive.
Because the simplification operator used in our algorithm is the edge collapse, during simplification, some holes in the mesh can be closed, but the topology is not changed.
Moreover, our algorithm works on both manifold and non-manifold meshes because of the simplification operator used.

A disadvantage of our method is complexity. The complexity is caused by the construction of the bounding volume hierarchy in the PQP library. For a local modification of the mesh, PQP has to rebuilt its bounding volume hierarchy in order to compute the distances between the meshes.
One perspective of this work is replacing the PQP library with a dynamic one. In this manner, a local modification of the mesh will not require the whole bounding volume hierarchy to be rebuilt.

Another perspective is improving vertex location. At this stage, the vertex obtained from an edge collapse is placed in so as to minimize the sum of quadratic distances computed as in the QEM algorithm. Because of approximations made by QEM in the computation of the geometric deviation the vertex is not placed in order to fit the best the geometry of the original mesh. In conclusion, we have to define a more accurate error metric for replacing the vertices.
The method for faces' subdivision could be also improved. Untill now, each face of the simplified mesh is equally subdivided, regardless of triangle's area. For a better approximation of the distance, one solution can be adapting the number of subdivision for a face to the face's area. In this manner, larger faces will more more subdivided while smaller ones will be lesser subdivided.
To improve the computational time of our algorithm, we propose to implement the algorithm on GPU.
In the future, we intend splitting the mesh into a number of unconnected regions and independently simplifying each region. The advantage of independent simplification is that the regions can be simplified in parallel to decrease the running time. A problem with independent simplification is the manner in which the regions are put back together. One approach is preserving unchanged the boundaries of each region. After the regions are reunited, the edges preserved unchanged on the boundaries are simplified.

Appendix

Appendix A

Appendix 1

2012 2nd International Conference on Image, Vision and Computing (ICIVC 2012)

IPCSIT vol. XX (2012) © (2012) IACSIT Press, Singapore

Mesh Simplification using a Two-Sided Error Minimization

Elena Ovreiu[1 2+], Juan Gabriel Riveros[2], Sebastien Valette[2], Rémy Prost[2]

[1] University Politehnica of Bucharest, Romania

[2] Université de Lyon, CREATIS; CNRS UMR5220; Inserm U1044; INSA-Lyon;

Université Lyon 1; France

Abstract. Reducing the complexity of a very large data set has become an important problem in the last years because of the rapid evolution of the acquisition techniques. In this paper we propose a mesh simplification algorithm based on two-sided error. The two-sided error metric permits us an accurately evaluation of the geometric deviation introduced by an edge simplification for the models with boundaries, islands.

Keywords: mesh simplification, two-sided error, quadratic error.

1. Introduction

Nowadays, meshes are presented in multiple and different areas such medical imaging, movie production, virtual reality, computer games. Due to the technological improvements from the recent years, a mesh could now have millions of elements. That means the reality could be reproduced more accurately with a complex mesh. The drawback of the complexity is the difficulty to manipulate this kind of meshes. For this reason, mesh simplification has become an extremely exploited topic in the last years.

The goal of mesh simplification is to reduce the complexity but keeping as possible as high fidelity of the original model.

Having this goal, a multitude of mesh simplification algorithms were developed during the time. For a detailed classification of those algorithms, we refer the reader to [4].

In the following we will describe some methods to compute the geometric deviation introduced by mesh simplification.

One of the most rapid simplification algorithms is the one proposed in [2]. The error introduced by one edge collapse is given by the sum of squared distances from the new vertex to its supporting planes. This error is called quadratic error metric (QEM). The drawback of this simplification method is given by the computed distance which is only an approximation. The algorithm presented in [5] is similar to QEM, but the error introduced by en edge collapse is considered the maximum of the distances from the resulted vertex to its supporting planes, and not the sum of them as in QEM.

In [6] the edge collapse is done accordingly to a complex error which represents the sum of four terms: the distance from the new vertex to the original model. This term penalizes contractions which do not preserve the sharp edges. The third term controls the accuracy of mesh's scalar attributes. The last term permits the optimization to get a desirable local minimum.

[+] Corresponding author: Tel: +33 667923072

E-mail address: Elena.ovreiu@gmail.com

2. Simplification Algorithm

Our simplification algorithm reduces the mesh complexity while retaining the mesh fidelity. The simplification is realized using an iteratively edge collapsing.The simplification algorithm follows the idea from [1].

The algorithm is outlined as follows:

1. On the original model we compute the error introduced by each possible edge collapse.
2. The edge with the minimum associated error is chosen to be collapsed.
3. We collapse the chosen edge to a single vertex $\boldsymbol{e} = (\boldsymbol{v_1}, \boldsymbol{v_2}) \rightarrow \overline{\boldsymbol{v}}$. The following operations are performed:
    - the position of the resulting vertex is computed: $(\boldsymbol{v_1}, \boldsymbol{v_2}) \rightarrow \overline{\boldsymbol{v}}$
    - the vertex v_2 and degenerated faces are eliminated;
    - all faces connected to $\boldsymbol{v_2}$ are connected to $\boldsymbol{v_1}$;
    - the error is recomputed for all edges in the new simplified model.
4. Those steps are repeated until the stop condition is achieved.

There is a main problem: how to define the error generated by the edge to be collapsed in order to keep a high fidelity to the original. We detail this problem in the following subsections.

In order to get an accurate measure of the error introduced by an edge collapse, we introduce a two sided quadratic error. For each possible edge contraction, we call *model state* the possible mesh configuration. That means for each possible edge collapse we get a possible model state and compute the quadratic error between this model and the original one and the reverse quadratic error (between the original model and possible model state).

2.1 Direct Error

We call *direct error* the quadratic error between the possible *model state* and the original model. We compute the distance from each face of the possible *model state* to the original model. The area weighted sum of the squared distances represents the error introduced by one edge collapse (eqn. 1).

$$d(\widehat{M}, M) = \frac{1}{\sum_{c \in \widehat{M}} w_c} \sum_{c \in \widehat{M}} w_c \, d^2(c, M) \quad (1)$$

where $\widehat{M}$ is the approximated model and M the original one.

In order to have more accurately measurement of the distance between two meshes, we apply a repetitive one-to-four subdivision *(1:4)* for each face (Fig.2). The number of subdivisions is variable for each face and it is chosen as so to have a proportion between the number of faces of the simplified mesh and the original one.

Thus, *d(c,M)* represents the distance from a cell (sub-triangle) of a subdivided triangle to the original mesh.

In practice, we compute the distance from a cell to the mesh as the arithmetic mean of the distances from the cell's vertices to the mesh:

$$d(c, M) = \frac{1}{3} \sum_{i=1}^{3} \mathrm{d}(\mathrm{v}_i, \mathrm{M}) \quad (2)$$

Where $d(v, M) = \min_{p \epsilon M, v \epsilon C} \|v - p\|$ is the minimum distance from the one vertex of subdivided cell to the closest face of M. $\|.\|$ is the Euclidean vector length operator.

w_c is a scalar weight factor which in our method is the area of a subdivided cell.

Like in [2], we weight the quadratic error by area in order to make simplification more robust to irregular sampling.

To reduce the complexity of computing the minimum Euclidean distance, we use the Proximity Query Package (PQP) library [3].

For each step of simplification, the errors for all edges are recomputed.

For one edge collapse, only the faces around the respective edge will be modified, so we recompute only the distances for those faces, and the other distances are not modified.

2.2 Reverse Error

We call the reverse error, the error between the original model and the simplified one. This error is similar to the direct one (equation 1), but here M and $\widehat{M}$ are interchanged.

For each possible edge contraction, for all faces, the distance from each face of the original model to the possible model state is modified. Computing all those distances is very expensive. Creating a copy of the possible model state in order to compute distances to this model is expensive as well. To avoid the computation of the distances for all faces in the original model, we compute only the distances for the faces in the original model which are affected by an edge collapse. For those faces in the simplified model, we look for the vertices in the original model for which the collapsed simulated faces are the closest. Afterwards, we compute the distances only for the faces in the original model which share those vertices. In this way we determine the faces on the original mesh whose distances are affected by one edge collapse.

To avoid recreating the simplified model for all possible edge collapses, we split the simplified model in more submodels, and we recreate only the submodel which contains the edge for which we compute the error.

2.3 Two-sided Error

The error for one edge is the sum of the direct error and the reverse one. Even if the direct error and the reverse one compute the distances between the same two models (possible model state and original one), these distances are not equal because they are computed in different directions.

The error associated to each edge is the global symmetric error between simplified model and original one. We use the term global because the error is computed from all the faces of one model to the other one. This global symmetric error gives us better approximations of the introduced geometric deviation, than a one sided error.

3. Results

The approximations obtained with our simplification algorithm are compared with the ones obtained with QEM [2]. We evaluate the error of approximations using quadratic error metric.

In Fig.1 Pieta is simplified up to 450 vertices using our simplification algorithm and QEM. We can see that for flat surfaces, our simplification algorithm uses fewer faces for the approximation.

lower than for approximations produced with QEM (Fig. 3). For instance, for an approximation of Pieta with 630 vertices, the quadratic error produced with our approximation algorithm is 0.088, while the one produced with QEM is 0.105. This difference is higher for a reduction factor higher than 50.

Fig. 4 represents an approximation for Octaflower model using our algorithm and QEM. We also compute the geometric deviations introduced by our simplification algorithm and by QEM using Metro tool [8]. For Octoflower model simplified up to 109 vertices (Fig.4) using our algorithm, the Hausdorff distance measured by Metro is 0.745652 while for the model simplified by QEM (with the same number of vertices), the Hausdorff distance is 1.355617. Also, the mean error is 0.0101324 (from the simplified model to original one) and 0.00382 (from the original to the simplified) for the model obtained with our algorithm while for the model simplified with QEM the mean errors are 0.130543 and 0.002525, respectively. For a drastic simplification (Octoflower with 39 vertices) we get the Hausdorff distance 0.215756 with our algorithm and 0.422858 with QEM. The mean errors are 0.29203 and 0.001669, respectively 0.035 and 0.001407 for QEM. For a Pieta model with 130 vertices, the Hausdorff distance (measured by Metro tool) is 7.288526 for the model simplified by our algorithm while on model simplified by QEM the Hausdorff distance is 19.107645. The mean error is 0.981812 (from simplified model to the original) and 0.06451 (the backward) for our simplification and 1.12817 and 0.056734 for QEM.

4. Conclusions

In conclusion, we propose an iterative edge collapsing algorithm which produces high quality approximations. We get high quality results using a two-sided error to characterize the geometric deviation introduced by a possible edge collapse. As, for each possible edge collapse, we compute the error between the whole simplified mesh and the original one, and the reverse error, between the original mesh and simplified one, we are able to perform an accurate characterization of the geometric error introduced by an edge collapse.

Even if we obtain better results than QEM in terms of quality of approximations, our algorithm is several times slower than QEM.

The simplification algorithm proposed in this paper gives us better approximations and better running time than the algorithm proposed in [1]. We obtain better approximations because, after each edge collapse, we recompute the error for all edges in the simplified mesh, and not only for the edges modified after one collapse.

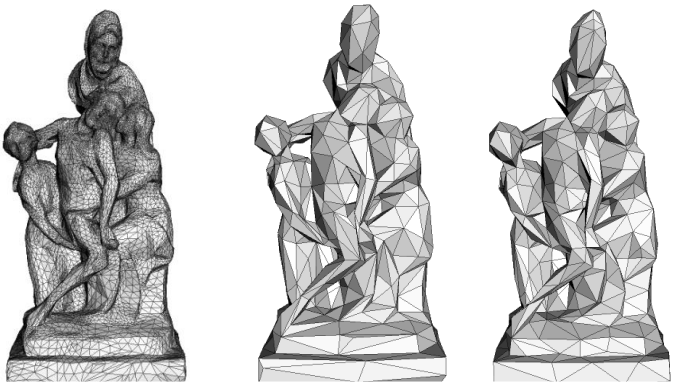


Fig. 1: Approximations of Pieta. From left to right: Original model with 13940 vertices (27904 triangles), simplified models with 450 vertices using our simplification algorithm and QEM.

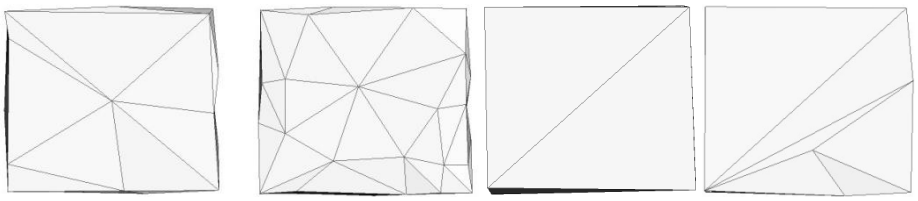


Fig. 2: Approximations for the base of Pieta

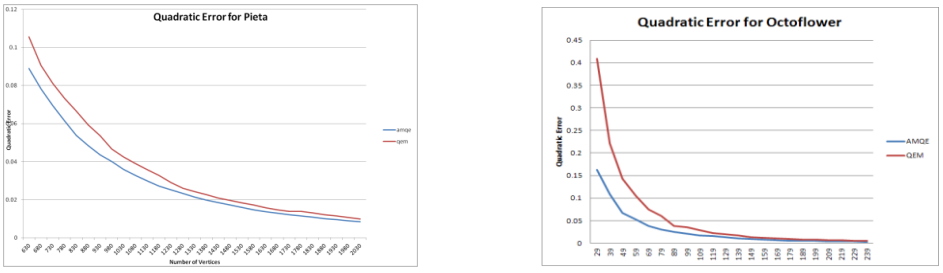


Fig 3: Geometric error vs. number of vertices for Pieta and Octoflower. The quadratic error is computed between approximated mesh and original one for each step of simplification. Simplifications are made using QEM (red line) and our algorithm (blue line).

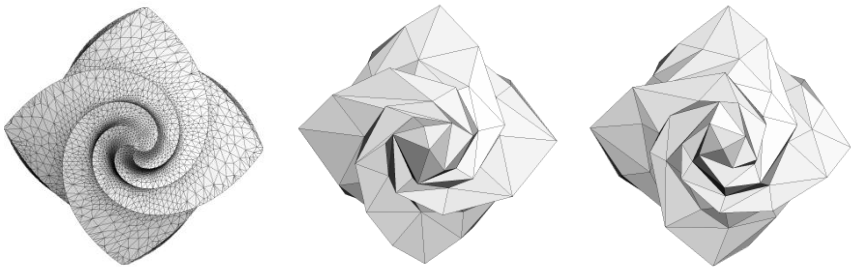


Fig.4: Approximations for Octoflower model**:** From left to right: Original model with 1919 vertices (15834 triangles), simplified models with 109 vertices using our simplification algorithm and QEM.

5. References

[1] M. Garland, P. Heckbert. *Surface Simplification Using Quadric Error*. In:.Proc. of ACM SIGGRAPH 1997, pp. 209-216.

[2] E. Larsen, S.Goottschalk, S. Lin and D.Manocha. *Rectangular Swept Sphere Volumes*. In:Proc. of IEEE Int. Conference on Robotics and Automation 2000.

[3] J.Talton. *A Short Survey of Mesh Simplification Algorithms*. Computers and Graphics, Elsevier, Volume 22, 2004.

[4] R. Ronfard, J. Rossignac. *Full-range approximations of triangulated polyhedra.* In: Proc. of Eurographics Vol. 15 ,1996.

[5] H. Hoppe. *Progressive meshes*. In: SIGGRAPH 96 Conference Proceedings, pages 99–108, 1996.

[6] R.Klein, G.Liebch, W. Strasser. *Mesh reduction with error control.* In: ACM Visualization 96, 1996.

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

Appendix B

Appendix 2

Mesh Simplification using an Accurate Measure of Quadratic Error

Elena Ovreiu*, Sebastien Valette†, Vasile Buzuloiu‡, Rémy Prost†
*INSA-Lyon, France,
University Politehnica of Bucharest,Romania,
Email: elena.ovreiu@creatis.insa-lyon.fr
†Université de Lyon, CREATIS; CNRS UMR5220; Inserm U1044; INSA-Lyon; Université Lyon 1; France.
Email: sebastien.valette@creatis.insa-lyon.fr, remy.prost@creatis.insa-lyon.fr
‡University Politehnica of Bucharest, Romania
Email: buzuloiu@alpha.imag.pub.ro

***Abstract*— In this paper we propose a new surface simplification algorithm which produces high quality approximations of the original models. The geometric simplification is based on iterative edge contractions. To get a simplified mesh which fairly fits the original one, we introduce an accurate measure of quadratic error to characterize the geometric deviation introduced by edge collapse. In addition, we propose a vertex optimization process which moves the new vertex towards an optimized position.**

I. Introduction

Reducing the complexity of geometric models has became a hot topic today due to the rapid improvements in the performance of acquisition techniques. The most common methods of mesh generation are 3D scanning, CT (computed tomography) and MRI (magnetic resonance imaging) scanners.The acquired geometric data are intensively used in applications like CAD (computer aided design), Movie Production, Computer Games, Medical Imaging, Virtual Reality. Large meshes (containing up to several millions of polygons) may slow the further computations done on them. Hence, between acquisition and production of geometric data, a processing step is necessary. It deals with approximation of a surface with another surface with fewer elements, in order to guarantee interactivity in 3D model rendering (the time to render a mesh is linear with the number of polygons), to eliminate redundant geometry for finite-element analysis or to reduce the model size, to improve the transmission over the Internet.

In this context, in the last period, a multitude of reducing complexity algorithms were developed. For a comparison between well-known simplification algorithms, we refer the reader to [4]. We will present some different methods to compute the error introduced by simplification. In [5] the function cost associated to a contraction is considered to be the maximum distance from the vertex resulted from an edge contraction to its supporting planes. Based on this method, in [1] a quadratic error metric (QEM) is proposed to compute the sum of squared distances from the new vertex to its planes. Although this algorithm is very fast and supports non-manifold models, the computed distance is an approximation of the true error and not an accurate one. In [6] the error is computed taking into consideration the geometric deviation introduced by an edge collapse, sharp edges and curvature preserving. [7] uses a more complex error criterion involving four terms: the distance from the new vertex to the original model, a representation term which penalizes contraction which do not preserve the mesh's sharp features, a metric which measures the accuracy of mesh's scalar attributes and a spring term which leads the optimization to a desirable local minimum. Computing the new position after contraction is a non-linear problem and for this reason the algorithm could be inefficient in practice. In [8], the priority queue is sorted accordingly to the Hausdorff distance. A vertex is removed from triangulation only if the deviation introduced by it is smaller than a predefined maximum Hausdorff distance.

Our main contribution consists in choosing an accurate quadratic error as the geometric deviation measure between the approximated mesh and the original one. With this geometric measure we build a priority queue, accordingly to the geometric simplification is performed. Also, we introduce an optimization process which minimizes the volume embedded between original mesh and simplified one [3]. More exactly, after an edge contraction $(v_1, v_2) \rightarrow \bar{v}$ is performed, we move the position of the new vertex $\bar{v}$ so that the deviation between faces adjacent to $\bar{v}$ and original mesh is minimized.

The paper is organized as follows: Section 2 presents an overview of the simplification algorithm, construction of the priority queue and optimization of the position of vertex obtained after collapsing one edge. The results are presented in Section 3 and, in Section 4 we draw some conclusions and future work.

II. Simplification Algorithm

A simplification algorithm takes as the input the original mesh and produces an approximation of this with fewer elements than the original. Also, the output should be a faithful approximation of the input. In our method, we use as input triangular meshes.

The simplification algorithm is based on iterative edge collapse and uses an accurate geometric deviation. After an edge contraction, an iterative optimal vertex placement is

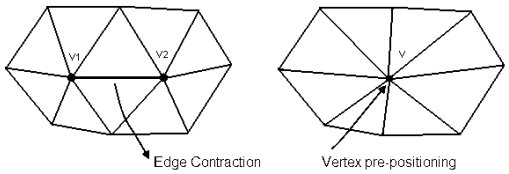


Fig. 1. **Edge collapse.** Vertices v_1 and v_2 are collapsed into a single vertex $\bar{v}$. The edge $e = (v_1, v_2)$ and the faces which share this edge are removed after contraction.

performed in order to obtain the best approximation of the original mesh. In order to permit a better characterization of the error introduced by a possible edge contraction, the resulted vertex should be placed in an optimal position. Thus, we pre-position the resulted vertex and after optimize its position. Our method is outlined as follows:

1) In the initial mesh, the cost associated to each edge contraction is computed.
2) Edges of the model are ordered by increasing cost in a priority queue.
3) The edge with the maximum priority is collapsed to a single vertex $e = (v_1, v_2) \rightarrow \bar{v}$. The following operations are performed:
   - a pre-position of the resulting vertex, $\bar{v}$ is computed: $(v_1, v_2 \rightarrow \bar{v})$.
   - the vertex is replaced to the optimal position.
   - the vertex v_2 and degenerated faces are eliminated;
   - all faces connected to v_2 are connected to v_1;
   - the priority queue is updated for all edges which previously were connected to v_1 and v_2;
4) Those steps are repeated until the desired number of vertices is achieved.

Moving the vertices v_1 and v_2 to a new position, the geometry of the mesh is modified. Also, the connectivity of the mesh is changed, connecting all edges which initially were connected to v_2, to v_1. Each edge contraction removes one vertex, three edges and two faces from the mesh. The priority queue is built accordingly to the function cost associated to each possible contraction. At a particular iteration, an error is associated with every possible contraction and the algorithm will apply the contraction with the minimum error. In our algorithm, the optimal vertex position is considered that position which minimizes the volume embedded between neighbouring region of the respective vertex and the original model. In the following we introduce the cost function used by our algorithm to measure the amount error introduce into approximation by each contraction.

A. Priority Queue

High quality approximations produced by a simplification algorithm depend on how the edge contractions are selected. In order to do this, the function cost associated to each contraction should characterize the geometric error introduced by that contraction as well as possible. In [1] the function cost is considered like being the sum of squared distances from the new vertex to its supporting planes. Computing the distance to the triangle's plane can underestimate the true error. We propose to introduce an error metric which is able to measure accurately the error introduced by an edge collapse, computing the distances from a point to a triangle, and not to its supporting plane.

We introduce a function cost (eq.1) which is the area-weighted sum of squared distances between the region modified by contraction and the original mesh:

$$E(M, \hat{M}) = \frac{1}{\sum_{c \in Supp(\hat{X})} w_c} \sum_{c \in Supp(\hat{X})} (w_c d^2(c, M)) + \frac{1}{\sum_{c \in T} w_c} \sum_{c \in T} (w_c d^2(c, \hat{M})) \quad (1)$$

where $Supp(\bar{v})$ is the region on $\hat{M}$ (approximated mesh) adjacent to the new vertex $\bar{v}$ and T represents the set of triangles on M (original mesh) where $Supp(\bar{v})$ is projected. In order to have a more accurate measurement of the distance between two meshes, we apply a one-to-four subdivision $(1 : 4)$ for each triangle (Fig.2). Thus, $d(c, M)$ represents the signed distance from a cell of a subdivided triangle to the original mesh. In practice, we compute the distance from a cell to the mesh as the arithmetic mean of the distances from the cell's vertices to the mesh:

$$d(c, M) = \frac{1}{3} \sum_{i=1}^{3} (d(v_i, M)) \quad (2)$$

where $d(v, M) = \min_{p \in M, v \in C} \|v - p\|$ is the minimum distance from the one vertex of subdivided cell to the closest face of M. $\|.\|$ is the Euclidian vector length operator. w_c is a scalar weight factor which in our method is the area of a subdivided cell. Like in [1], we weight the quadratic error by area in order to achieve an error independent by mesh tessellation. To reduce the complexity of computing the minimum Euclidian distance, we use the Proximity Query Package (PQP) library [2].

The number of subdivisions is computed using the following formula:

$$N_{subd} = floor(0.5 + log(N2/N1)/log(4.0)) \quad (3)$$

where N_1 and N_2 represents the number of faces of the approximated mesh, and of the original mesh. To get the coordinates of the sampling points for each cell in the mesh, in our implementation, we use a general subdivided triangle which has as vertices coordinates its barycentric coordinates. Thus, each sample point will have as coordinates its barycentric coordinates. This general subdivided triangle is matched on each mesh's cell, and the sampling points coordinates for the respective cell are computed as linear interpolation between cells's vertices and coordinates of points of the general subdivided triangle.

B. Optimization of Vertex Position

When an edge is merged to a single vertex, $(v_1, v_2) \rightarrow \bar{v}$ an important aspect is to place the resulted vertex in a position which best fits the original model. In our algorithm, we initially place the vertex $\bar{v}$ using the method proposed in [1] and after that, we minimize the volume embedded between simplified mesh and original one. In [1], the position for $\bar{v}$ is that which minimizes $\Delta(\bar{v})$, where $\Delta(\bar{v}) = \bar{v}^T \bar{Q} \bar{v}$, $\bar{Q}$ being the quadratic error metric associated to $\bar{v}$. For each edge cost computation, firstly we pre-position the vertex the respective edge could merge to and move vertex in order to find the local optimum. The local optimum is considered the position for $\bar{v}$ where the volume between the region surrounding $\bar{v}$ and original mesh is minimized. To minimize the error, it is necessary to define the geometric deviation between two 3D surfaces. In our method, the geometric deviation is considered to be the volume embedded between simplified mesh and the original one. In order to minimize the volume between the faces adjacent to that vertex and original mesh, and implicitly to reduce the quadratic error between approximated mesh and original one, we introduce the following formula:

$$disp(X_i) = \frac{\sum_{c \in Supp(X_i)} \lambda_c w_c d(c, M)}{\sum_{c \in Supp(X_i)} \lambda_c w_c} \quad (4)$$

$disp(X_i)$ is the displacement for vertex X_i and $Supp(X_i)$ denotes the adjacent faces of X_i. The direction of the displacement is given by the vectorial sum of the distances between $\hat{M}$ and M. The magnitude of the movement is given by the sum of the distances from the subdivided cells to the original model. The sum is scaled by the subdivided cells areas, and by the shape function λ_c. λ_c is set to 1 on the X_i and decreases to 0 towards the neighbouring vertices (Fig.2). For a point in the interior of a triangle, we deduce the value of the function shape using barycentric coordinates of the respective point. Thus, if the point has the barycentric coordinates $(\lambda_1, \lambda_2, \lambda_3)$, the shape function for the respective point will be λ_1. The shape function of a cell is the arithmetic mean of the shape functions of its vertices.

In practice, we get the magnitude and the direction of the distances using PQP library. The evolution of the vertex position, for the k^{th} step can be written as:

$$X_i^{k^{th}} = X_i^{k^{th}-1} + disp(X_i^{k^{th}-1}) \quad (5)$$

After each optimization step, the volume embedded between the approximated mesh and the original one will be reduced. Therefore, the quadratic error and the Hausdorff distance between those meshes decreases (Fig.3)

III. Results

We evaluate the quality of approximations produced by our algorithm using the error from equation 1. Also, the quality of our method is evaluated with the Hausdorff distance.

Comparing our algorithm with Quadratic Error Metric proposed in [1] in terms of quadratic distance between approximations and original models, our algorithm produces better

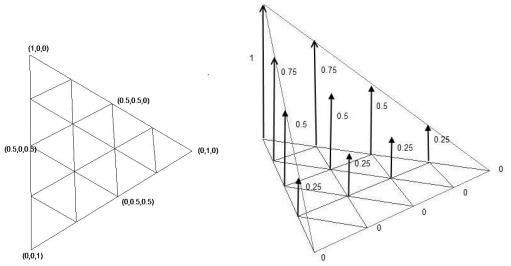


Fig. 2. **The shape functions.** The shape function of a point inside of a face is determined from barycentric coordinates of the respective point.

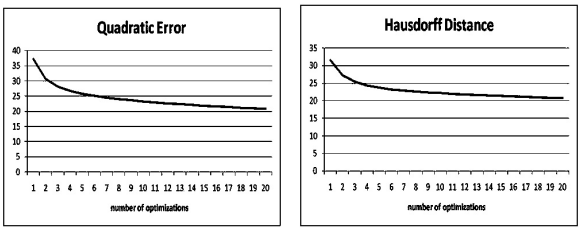


Fig. 3. **Quadratic Error and Hausdorff distance vs. number of optimizations.** The Heart model is approximated with 30 vertices. The position of each vertex in the approximation is optimized using 20 optimization steps .

results for a reduction bigger than 50% (Fig.4). For instance, for an approximation with 125 vertices, the quadratic error introduced by our algorithm is 4.88 while by QEM is 5.12. For an approximation using 50 vertices, the error introduced by our algorithm is 22.98 while by QEM is 32.25 or 37.98, respectively 58.06 for an approximation with 37 vertices. Regarding the processing time, our algorithm is slower than [1].

Figure 5 shows a sequence of approximations using our algorithm. We can observe that during the simplification, the major details like horns, tail remain. They are altered for drastic complexity reductions (simplified model with 50 vertices). Figures 5d shows the model simplified without optimization the position of vertices while in Figure 5c during mesh simplification, a vertex position optimization is performed. For the vertex optimization process, we fixed 5 steps of optimization. We can see that optimal vertex movement produces well-shaped models. In Figure 6 a sphere with 2000 vertices is simplified obtaining a set of approximations with 500, 100 and 50 vertices. In Figure 7 we simplify a genus 3 model and we can observe that the topology is preserved during the simplification using our accurate error metric.

IV. Conclusion and Future Work

We have presented an iteratively edge collapsing algorithm which produces high quality approximations of original models.There were two contributions which lead to those results: the accurate measurement of the geometric deviation introduced by each contraction operation and the positioning of the vertex resulted from an edge collapse. In our algorithm, the collapsing cost is given by the area weighted sum of

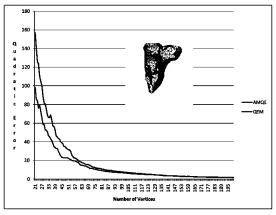


Fig. 4. **Geometric error vs. number of vertices.** The quadratic error is computed between approximated mesh and original one for each step of simplification. Simplifications are made using QEM (red line) and our algorithm (blue line) on the heart model

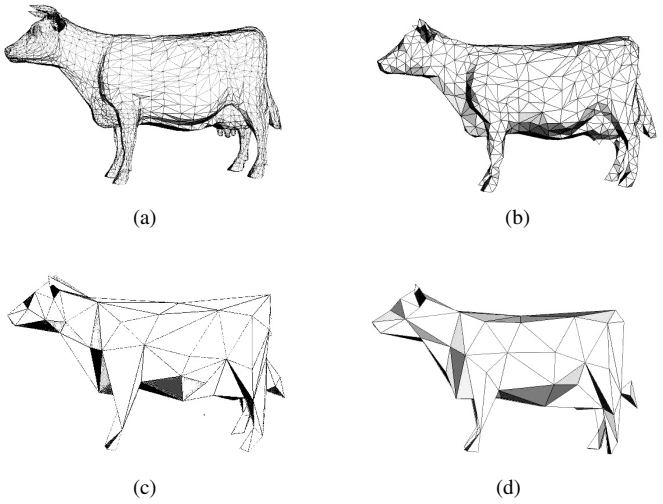


Fig. 5. **A sequence of approximations using our algorithms.** From left to right:Original model with 2903 vertices (5804 triangles). Simplified models with 1000,700,300 and respectively 50 vertices.

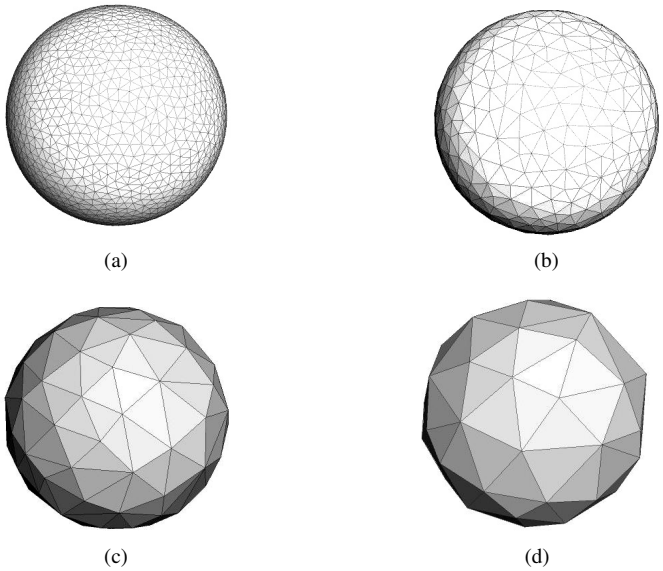


Fig. 6. **Approximations of a triangulated sphere.** Original model with 2000 vertices. Approximation models using 500, 100 and 50 vertices.

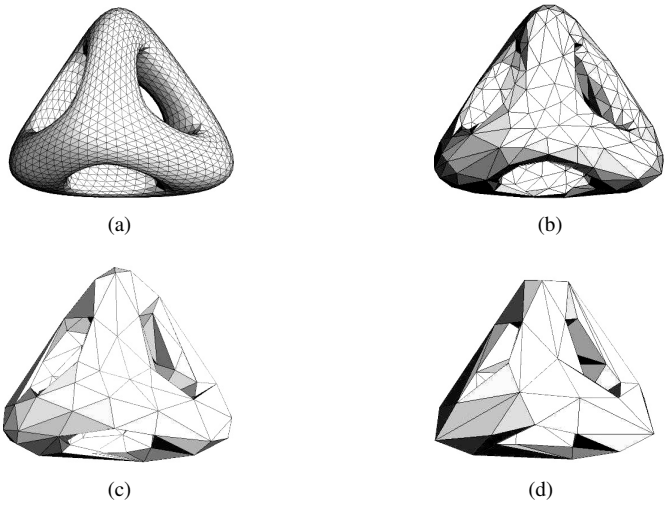


Fig. 7. **Approximations of a genus 3.** Original model with 2000 vertices. Approximation models using 500, 100 and 50 vertices.

squared distances between the region surrounding the new vertex and original model. Because we compute the distance to the triangles of the original mesh and not to the planes of triangles it makes this algorithm accurately. Also, our algorithm computes the distance between two models in a symmetric fashion: from triangles adjacent to the new vertex to the original mesh, and from the triangles on original mesh where the first distances are projected to the current mesh. The computation of error is complex but the advantage is it is very accurate. Another factor which leads to the accuracy of the method is the optimization of the position of vertex resulted from edge collapse. We move the vertex position in order to minimize the volume embedded between simplified mesh and approximated one. As future work, we intend to simplify the computation complexity for our accurate quadratic error, making the algorithm faster. Regarding the vertex optimization process, we are working on finding the optimal vertex position using fewer iterations.

ACKNOWLEDGMENT

This work was supported in part by the Region Rône-Alpes Cluster 2ISLE, PP3, subproject SIMED.

REFERENCES

[1] M. Garland and P. Heckbert, *Surface Simplification Using Quadric Error, Proc. of ACM SIGGRAPH, 209-216, 1997.*

[2] E. Larsen, S. Goottschalk, S. Lin and D. Manocha, *Distance Queries with Rectangular Swept Sphere Volumes, Proc. of IEEE Int. Conference on Robotics and Automation,2000.*

[3] P. Alliez, N.Laurent, H. Sanson, F. Schmitt,*Mesh Approximation using a Volume-Based Metric , Proc. of IEEE Seventh Pacific Conference,1999.*

[4] J. Talton, *A Short Survey of Mesh Simplification Algorithms , Computers and Graphics, Elsevier, Volume 22,,2004.*

[5] R. Ronfard, J. Rossignac.*Full-range approximations of triangulated polyhedra. Proc. of Eurographics Vol. 15 ,1996.*

[6] S-J. Kim,S-K. Kim,C-H. Kim. *Discrete Differential Error Metric for Surface Simplification. In Pacific Graphics, pages 276–283,2002.*

[7] H. Hoppe.*Progressive meshes. SIGGRAPH 96 Conference Proceedings, pages 99108,1996.*

Bibliography

[Alliez *et al.*, 1999] Alliez, P., Laurent, N., Sanson, H., and Schmitt, F. (1999). Mesh approximation using a volume-based metric. *Computer Graphics and Applications, Pacific Conference on*, 0:292.

[Bajaj, 1996] Bajaj, C. L. (1996). Error-bounded reduction of triangle meshes with multivariate data. pages 34–45.

[Bartoň *et al.*, 2010] Bartoň, M., Hanniel, I., Elber, G., and Kim, M.-S. (2010). Precise hausdorff distance computation between polygonal meshes. *Comput. Aided Geom. Des.*, 27(8):580–591.

[Bernardini *et al.*, 2002] Bernardini, F., Rushmeier, H., Martin, I. M., Mittleman, J., and Taubin, G. (2002). Building a digital model of michelangelo's florentine pietÃă. *IEEE Computer Graphics and Applications*, 22:59–67.

[Blythe, 2006] Blythe, D. (2006). The direct3d 10 system. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 724–734, New York, NY, USA. ACM.

[Borodin *et al.*, 2003] Borodin, P., Gumhold, S., Guthe, M., and Klein, R. (2003). High-quality simplification with generalized pair contractions. In *In GraphiCon 2003*, pages 147–154.

[Borouchaki and Frey, 2005] Borouchaki, H. and Frey, P. (2005). Simplification of surface mesh using hausdorff envelope. *Computer Methods in Applied Mechanics and Engineering*, 194(48-49):4864–4884.

[Botsch *et al.*, 2008] Botsch, M., Pauly, M., Kobbelt, L., Alliez, P., and uno LÃľvy, B. (2008). Geometric modeling based on polygonal meshes. In *Eurographics Tutorial*.

[Botsch *et al.*, 2006] Botsch, M., Pauly, M., Rossl, C., and S., B. (2006). *Geometric Modeling Based on Triangle Meshes*.

[Ciampalini *et al.*, 1997] Ciampalini, A., Cignoni, P., Montani, C., and Scopigno, R. (1997). Multiresolution decimation based on global error. *The Visual Computer*, 13:228–246. 10.1007/s003710050101.

[Cignoni *et al.*, 1996] Cignoni, P., Rocchini, C., and Scopigno, R. (1996). Metro: measuring error on simplified surfaces. Technical report, Paris, France, France.

[Clark, 1976] Clark, J. H. (1976). Seminal graphics. chapter Hierarchical geometric models for visible surface algorithms, pages 43–50. ACM, New York, NY, USA.

[Cohen *et al.*, 1998] Cohen, J., Olano, M., and Manocha, D. (1998). Appearance-preserving simplification. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 115–122, New York, NY, USA. ACM.

[Cohen *et al.*, 1996] Cohen, J., Varshney, A., Manocha, D., Turk, G., Weber, H., Agarwal, P., Brooks, F., and Wright, W. (1996). Simplification envelopes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 119–128, New York, NY, USA. ACM.

[Cohen-Steiner *et al.*, 2004] Cohen-Steiner, D., Alliez, P., and Desbrun, M. (2004). Variational shape approximation. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 905–914, New York, NY, USA. ACM.

[DeCoro and Tatarchuk, 2007] DeCoro, C. and Tatarchuk, N. (2007). Real-time mesh simplification using the gpu. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, I3D '07, pages 161–166, New York, NY, USA. ACM.

[Digne *et al.*, 2012] Digne, J., Cohen-Steiner, D., Alliez, P., Desbrun, M., and De Goes, F. (2012). Feature-Preserving Surface Reconstruction and Simplification from Defect-Laden Point Sets. Rapport de recherche RR-7991, INRIA.

[Division and Guéziec, 1996] Division, I. B. M. C. R. and Guéziec, A. (1996). *Surface Simplification Inside a Tolerance Volume*. Research report. IBM T.J. Watson Research Center.

[Division *et al.*, 1993] Division, I. B. M. C. R., Kalvin, A., and Taylor, R. (1993). *Super-Faces: Polyhedral Approximation with Bounded Error*.

[El-Sana and Varshney, 1997] El-Sana, J. and Varshney, A. (1997). Controlled simplification of genus for polygonal models.

[Erikson and Manocha, 1999] Erikson, C. and Manocha, D. (1999). Gaps: General and automatic polygonal simplification.

[Farin, 2002] Farin, G. (2002). *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition.

[Frisken *et al.*, 2000] Frisken, S. F., Frisken, S. F., Perry, R. N., Perry, R. N., Rockwood, A. P., Rockwood, A. P., Jones, T. R., and Jones, T. R. (2000). Adaptively sampled distance fields: A general representation of shape for computer graphics. pages 249–254.

[Garland, 1999] Garland, M. (1999). *Quadric-based polygonal surface simplification*. PhD thesis, Pittsburgh, PA, USA. AAI9950005.

[Garland and Heckbert, 1997] Garland, M. and Heckbert, P. S. (1997). Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 209–216, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

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

[Garland and Zhou, 2005] Garland, M. and Zhou, Y. (2005). Quadric-based simplification in any dimension. *ACM Transactions on Graphics*, 24:209–239.

[Gottschalk *et al.*, 1996] Gottschalk, S., Lin, M. C., and Manocha, D. (1996). Obb-tree: A hierarchical structure for rapid interference detection.

[Hamann, 1994] Hamann, B. (1994). A data reduction scheme for triangulated surfaces. *Comput. Aided Geom. Des.*, 11(2):197–214.

[Hamann and Chen, 1994] Hamann, B. and Chen, J.-L. (1994). Data point selection for piecewise linear curve approximation. *Computer Aided Geometric Design*, 11(3):289–301.

[Heckbert and Garland, 1995] Heckbert, P. S. and Garland, M. (1995). Survey of polygonal surface simplification algorithms.

[Heckbert and Garland, 1999] Heckbert, P. S. and Garland, M. (1999). Optimal triangulation and quadric-based surface simplification. *Comput. Geom. Theory Appl.*, 14(1-3):49–65.

[Hinker and Hansen, 1993] Hinker, P. and Hansen, C. (1993). Geometric optimization. In *Proceedings of the 4th conference on Visualization '93*, VIS '93, pages 189–195, Washington, DC, USA. IEEE Computer Society.

[Ho *et al.*, 2001] Ho, J., Lee, K.-C., and Kriegman, D. (2001). Compressing large polygonal models. In *SIGGRAPH Technical Sketch.*

[Hoppe, 1996] Hoppe, H. (1996). Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 99–108, New York, NY, USA. ACM.

[Hoppe, 1997] Hoppe, H. (1997). View-dependent refinement of progressive meshes. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 189–198, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[Hoppe, 1998] Hoppe, H. (1998). Smooth view-dependent level-of-detail control and its application to terrain rendering. In *Proceedings of the conference on Visualization '98*, VIS '98, pages 35–42, Los Alamitos, CA, USA. IEEE Computer Society Press.

[Hoppe, 1999] Hoppe, H. (1999). New quadric metric for simplifying meshes with appearance attributes.

[Hoppe *et al.*, 1993] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. (1993). Mesh optimization. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, pages 19–26, New York, NY, USA. ACM.

[Hoppe and Wa, 1998] Hoppe, H. and Wa, R. (1998). Efficient implementation of progressive meshes.

[Isenburg *et al.*, 2003] Isenburg, M., Lindstrom, P., Gumhold, S., and Snoeyink, J. (2003). Large mesh simplification using processing sequences. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, pages 61–, Washington, DC, USA. IEEE Computer Society.

[Kalvin and Taylor, 1996] Kalvin, A. D. and Taylor, R. H. (1996). Superfaces: Polygonal mesh simplification with bounded error. *IEEE Comput. Graph. Appl.*, 16(3):64–77.

[Klein *et al.*, 1996] Klein, R., Liebich, G., and StraÃ§er, W. (1996). Mesh reduction with error control. In *Visualization 96. ACM*, pages 311–318.

[Kobbelt *et al.*, 1998] Kobbelt, L., Campagna, S., and peter Seidel, H. (1998). A general framework for mesh decimation. In *in Proceedings of Graphics Interface*, pages 43–50.

[Larsen *et al.*, 1999] Larsen, E., Gottschalk, S., Lin, M. C., and Manocha, D. (1999). Fast proximity queries with swept sphere volumes. Technical report.

[Larsen *et al.*, 2000] Larsen, E., Gottschalk, S., Lin, M. C., and Manocha, D. (2000). Fast distance queries with rectangular swept sphere volumes. In *Proc. of IEEE Int. Conference on Robotics and Automation*, pages 3719–3726.

[Lindstrom, 2000] Lindstrom, P. (2000). Out-of-core simplification of large polygonal models.

[Lindstrom and Silva, 2001] Lindstrom, P. and Silva, C. T. (2001). A memory insensitive technique for large model simplification. In *Proceedings of the conference on Visualization '01*, VIS '01, pages 121–126, Washington, DC, USA. IEEE Computer Society.

[Lindstrom and Turk, 2000] Lindstrom, P. and Turk, G. (2000). Image-driven simplification. *ACM Trans. Graph.*, 19(3):204–241.

[Lorensen and Cline, 1987] Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. *COMPUTER GRAPHICS*, 21(4):163–169.

[Low and Tan, 1997] Low, K.-L. and Tan, T.-S. (1997). Model simplification using vertex-clustering. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, I3D '97, pages 75–ff., New York, NY, USA. ACM.

[Luebke, 1996] Luebke, D. (1996). Hierarchical structures for dynamic polygonal simplification. Technical report.

[Luebke and Erikson, 1997] Luebke, D. and Erikson, C. (1997). View-dependent simplification of arbitrary polygonal environments. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 199–208, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[Mäntylä, 1987] Mäntylä, M. (1987). *An introduction to solid modeling*. Computer Science Press, Inc., New York, NY, USA.

[Marinov and Kobbelt, 2005] Marinov, M. and Kobbelt, L. (2005). Automatic generation of structure preserving multiresolution models.

[Meftah *et al.*, 2010] Meftah, A., Roquel, A., Payan, F., and Antonini, M. (2010). Measuring Errors for Massive Triangle Meshes. In IEEE, editor, *Proceedings of IEEE international workshop on MultiMedia Signal Processing (MMSP)*, page 5, France.

[Prince, 2000] Prince, C. (2000). Progressive meshes for large models of arbitrary topology. Technical report.

[Ramer *et al.*, 1972] Ramer, U., ENGINEERING, N. Y. U. B. D. O. E., and SCIENCE., C. (1972). *An Iterative Procedure for the Polygonal Approximation of Plane Curves*. Defense Technical Information Center.

[Ronfard *et al.*, 1996] Ronfard, R., Rossignac, J., and Rossignac, J. (1996). Full-range approximation of triangulated polyhedra.

[Rossignac and Borrell, 1992] Rossignac, J. and Borrell, P. (1992). *Multi-resolution 3D Approximations for Rendering Complex Scenes*. Research report // IBM Watson Research Center. IBM Research Division, T. J. Watson Research Center.

[Samet, 1990] Samet, H. (1990). *The design and analysis of spatial data structures*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[Schroeder, 1997] Schroeder, W. J. (1997). A topology modifying progressive decimation algorithm. In *Proceedings of the 8th conference on Visualization '97*, VIS '97, pages 205–ff., Los Alamitos, CA, USA. IEEE Computer Society Press.

[Schroeder *et al.*, 1992] Schroeder, W. J., Zarge, J. A., and Lorensen, W. E. (1992). Decimation of triangle meshes. *SIGGRAPH Comput. Graph.*, 26(2):65–70.

[Shaffer and Garland, 2001] Shaffer, E. and Garland, M. (2001). Efficient adaptive simplification of massive meshes. In *IEEE Visualization*.

[Soucy and Laurendeau, 1992] Soucy, M. and Laurendeau, D. (1992). Multi-resolution surface modeling from multiple range view. In *Proc. CVPR'92*, pages 348–353, Champaign, IL.

[Soucy and Laurendeau, 1996] Soucy, M. and Laurendeau, D. (1996). Multiresolution surface modeling based on hierarchical triangulation. *Computer Vision and Image Understanding*, 63:1–14.

[Straub, 2007] Straub, R. (2007). Exact Computation of the Hausdorff Distance between Triangular Meshes . pages 17–20.

[Tang *et al.*, 2009] Tang, M., Lee, M., and Kim, Y. J. (2009). Interactive hausdorff distance computation for general polygonal models. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2009)*, 28(3):to appear.

[Turk, 1992] Turk, G. (1992). Re-tiling polygonal surfaces. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '92, pages 55–64, New York, NY, USA. ACM.

[Valette and Chassery, 2004] Valette, S. and Chassery, J. (2004). Approximated centroidal voronoi diagrams for uniform polygonal mesh coarsening. *Computer Graphics Forum (Eurographics 2004 proceedings)*, 23(3):381–389.

[Valette *et al.*, 2005] Valette, S., Kompatsiaris, I., and Chassery, J. (2005). Adaptive polygonal mesh simplification with discrete centroidal voronoi diagrams. In *Proceedings of 2nd International Conference on Machine Intelligence ICMI 2005*, pages 655–662, Tozeur, Tunisia.

[Varshney, 1994] Varshney, A. (1994). Hierarchical geometric approximations.

[Varshney *et al.*, 1995] Varshney, A., Agarwal, P. K., Brooks, F. P., Jr., Wright, W. V., and Weber, H. (1995). Generating levels of detail for large-scale polygonal models. Technical report.

[Wu and Kobbelt, 2003a] Wu, J. and Kobbelt, L. (2003a). Piecewise linear approximation of signed distance fields.

[Wu and Kobbelt, 2003b] Wu, J. and Kobbelt, L. (2003b). A stream algorithm for the decimation of massive meshes.

[Zelinka, 2002] Zelinka, S. D. (2002). Permission grids: Practical, error-bounded simplification. *ACM Transactions on Graphics*, 21:2002.

[Zorin and Schröder, 2000] Zorin, D. and Schröder, P. (2000). *Subdivision for Modeling and Animation: SIGGRAPH 2000 Course Notes.*

Personal Bibliography

Journals

R. Poranne, E. Ovreiu, C. Gotsman, **Interactive Planarization and Optimization for 3D Meshes**, *Computer Graphics Forum* (accepted).

E. Ovreiu, S. Valette, R. Prost, **Triangular Mesh Simplification using a Symmetric Accurate Error Metric**, *Computer & Graphics* (to be submitted)

International Conferences

E. Ovreiu, J.G. Riveros, S. Valette, R. Prost, **Mesh Simplification using a Two-Sided Error Minimization**, *ICVIC 2012*, Shanghai, China, 08/2012

E. Ovreiu, S. Valette, V. Buzuloiu, R. Prost, **Mesh Simplification using an Accurately Measured Quadratic Error**, *IEEE, International Symposium on Signals, Circuits & Systems, ISSCS 2011*, Iasi, Romania, pp. 39-42, 06/2011

Oral Communications

E. Ovreiu, R. Poranne, C. Gotsman, **Modeling Meshes with Planar Faces**, *XIème Colloque Franco-Roumain de Mathématiques Appliquées*, Bucharest, Romania, 08/2012.

TITRE EN FRANCAIS

Simplification précise de maillages 3D

RESUME EN FRANCAIS

Les objets numériques 3D sont utilisés dans de nombreux domaines, les films d'animations, la visualisation scientifique, l'imagerie médicale, la vision par ordinateur.... Ces objets sont généralement représentés par des maillages à faces triangulaires avec un nombre énorme de triangles. La simplification de ces objets, avec préservation de la géométrie originale, a fait l'objet de nombreux travaux durant ces dernières années.

Dans cette thèse, nous proposons un algorithme de simplification qui permet l'obtention d'objets simplifiés de grande précision. Nous utilisons des fusions de couples de sommets avec une relocalisation du sommet résultant qui minimise une métrique d'erreur. Nous utilisons deux types de mesures quadratiques de l'erreur : l'une uniquement entre l'objet simplifié et l'objet original (Accurate Measure of Quadratic Error (AMQE)) et l'autre prend aussi en compte l'erreur entre l'objet original et l'objet simplifié ((Symmetric Measure of Quadratic Error (SMQE)) . Le coût calculatoire est plus important pour la seconde mesure mais elle permet une préservation des arêtes vives et des régions isolées de l'objet original par l'algorithme de simplification. Les deux mesures conduisent à des objets simplifiés plus fidèles aux originaux que les méthodes actuelles de la littérature.

TITRE EN ANGLAIS

Accurate 3D Mesh Simplification

RESUME EN ANGLAIS

Complex 3D digital objects are used in many domains such as animation films, scientific visualization, medical imaging and computer vision. These objects are usually represented by triangular meshes with many triangles. The simplification of those objects with the target to keep them as close as possible to the original ones has received a lot of attention in the last years.

In this context, we propose a simplification algorithm which is focused on the accuracy of the simplifications. The mesh simplification uses edges collapses with vertex relocation by minimizing an error metric. Accuracy is obtained with the two error metrics we use: the Accurate Measure of Quadratic Error (AMQE) and the Symmetric Measure of Quadratic Error (SMQE).

AMQE is computed as the weighted sum of squared distances between the simplified mesh and the original one. Accuracy of the measure of the geometric deviation introduced in the mesh by an edge collapse is given by the distances between surfaces. The distances are computed in between sample points of the simplified mesh and the faces of the original one.

SMQE is similar to the AMQE method but computed in the both, direct and reverse directions, i.e. simplified to original and original to simplified meshes. The SMQE approach is computationnaly more expensive than the AMQE but the advantage of computing the AMQE in a reverse way results in the preservation of boundaries, sharp features and isolated regions of the mesh.

For both measures we obtain better results than methods proposed in the literature.

MOTS-CLES

mesh simplification, edge collapse, accurate metric, symmetric metric, quadratic error

INTITULE ET ADRESSE DE L'U.F.R. OU DU LABORATOIRE

Université de Lyon, CREATIS ; CNRS UMR5220 ; Inserm U1044 ; INSA-Lyon ; Université Lyon 1, 7 Av. Jean Capelle, 69621 VILLLEURBANNE, France.