A PARTITION BASED METHOD FOR SPECTRUM-PRESERVING MESH SIMPLIFICATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MİSRANUR YAZGAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

AUGUST 2022

Approval of the thesis:

A PARTITION BASED METHOD FOR SPECTRUM-PRESERVING MESH SIMPLIFICATION

submitted by **MİSRANUR YAZGAN** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Assoc. Prof. Dr. Yusuf Sahillioğlu
Supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Yücel Yemez
Computer Engineering, Koç University _____

Assoc. Prof. Dr. Yusuf Sahillioğlu
Computer Engineering, METU _____

Assist. Prof. Dr. Hakan Yıldız
Computer Engineering, METU _____

Date: 29.08.2022

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Misranur Yazgan

Signature :

ABSTRACT

A PARTITION BASED METHOD FOR SPECTRUM-PRESERVING MESH SIMPLIFICATION

Yazgan, Misranur
M.S., Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. Yusuf Sahillioğlu

August 2022, 56 pages

When the complexity of a mesh starts introducing high computational costs, mesh simplification methods come into the picture, to reduce the number of elements utilized to represent the mesh. Majority of the simplification methods focus on preserving the appearance of the mesh, ignoring the spectral properties of the differential operators derived from the mesh. The spectrum of the Laplace-Beltrami operator is essential for a large subset of applications in geometry processing. Coarsening a mesh without considering its spectral properties might result with incorrect calculations on the simplified mesh. Given a 3D triangular mesh, this thesis aims to decrease its resolution by applying mesh simplification, while focusing on preserving the spectral properties of the associated cotangent Laplace-Beltrami operator. Unlike the existing spectrum-preserving coarsening methods, this work utilizes solely the eigenvalues of the operator, in order to preserve the spectrum. The presented method is partition based, in a way that the input mesh is divided into smaller patches and each patch is simplified individually. The method is evaluated on a variety of meshes, by using functional maps and quantitative norms. These metrics are used to measure how well the eigenvalues and eigenvectors of the Laplace-Beltrami operator computed on the

input mesh are maintained by the output mesh. At the end of this thesis, it is demonstrated that the achieved spectrum preservation is at least as effective as the existing spectral coarsening methods.

Keywords: Laplace-Beltrami operator, mesh simplification, spectral mesh simplification, spectral coarsening, geometry processing

ÖZ

SPEKTRUM KORUMA ODAKLI POLİGONAL BASİTLEŞTİRME İÇİN BÖLÜMLEYİCİ BİR YÖNTEM

Yazgan, Misranur
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Yusuf Sahillioğlu

Ağustos 2022 , 56 sayfa

Yüksek çözünürlüğe sahip poligonal modeller yüksek hesaplama maliyetine sebep olmaya başladıklarında, modeli oluşturan eleman sayısını azaltmak için poligonal basitleştirme metodlarına ihtiyaç duyulur. Poligonal basitleştirme algoritmalarının büyük bir çoğunluğu modelin görüntüsünü koruma odaklıdır ve poligonal model için oluşturulan diferansiyel operatörlerin spektral özelliklerini göz ardı ederler. Laplace-Beltrami operatörünün spektrumu, geometri işleme uygulamalarının geniş bir alt kümesi için gerekli temel bir bileşendir. Bir modeli spektral özelliklerini dikkate almadan basitleştirmek, basitleştirilmiş model üzerinde yapılan hesaplamaların yanlış olmasına sebep olabilir. Bu tezin amacı, verilen 3B modelin çözünürlüğünü poligonal basitleştirme yöntemleri ile düşürürken, model için oluşturulan kotanjant Laplace-Beltrami operatörünün spektral özelliklerini korumaktır. Mevcut spektrum koruma odaklı basitleştirme yöntemlerinin aksine, bu çalışmada spektrumu korumak için yalnızca operatörün özdeğerleri kullanılmaktadır. Bu tezde önerilen yöntem, modeli daha küçük bölümlere ayırıp, basitleştirme yöntemini her bir bölüm için ayrı ayrı uygulama üzerine kuruludur. Yöntem, farklı modeller üzerinde denenmiş ve fonksiyonel

eşleme ve nicel normlar ile değerlendirilmiştir. Bu ölçütler, girdi olarak verilen model üzerinde hesaplanan Laplace-Beltrami operatörünün özdeğerlerinin ve özvektörlerinin, çıktı model tarafından ne kadar iyi korunduğunu ölçmek için kullanılır. Bu tezin sonunda, elde edilen spektrum korumanın, en az mevcut spektrum koruma odaklı basitleştirme yöntemleri kadar etkili olduğu gösterilmiştir.

Anahtar Kelimeler: Laplace-Beltrami operatörü, poligonal basitleştirme, spektral poligonal basitleştirme, spektrum koruma, geometri işleme

ACKNOWLEDGEMENTS

Firstly, I would like to sincerely thank my thesis advisor Assoc. Prof. Dr. Yusuf Sahillioğlu for his invaluable guidance, understanding and friendly attitude through this study. It is indisputable that his vision and collaboration helped me very much, putting together this work.

I would like to thank my family for providing me their continuous support and love, throughout my whole life. I am grateful for their faith in me, their appreciation for what I do and for the motivation they provided even from far away.

I would also like to thank my close friends here and overseas, who has always been motivating and encouraging me through the process.

And lastly, my special thanks goes to my love Furkan, who has been standing by my side all through this journey, never once wavering in giving me his endless support, giving me valuable advice and endless motivation. It would not have been possible for me to make it through this, without him glued to the desk with me, side by side.

To my family and my love

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS ix

TABLE OF CONTENTS xi

LIST OF FIGURES xiii

LIST OF TABLES xvi

LIST OF ABBREVIATIONS xvii

CHAPTERS

1 INTRODUCTION 1

1.1 Motivation and Problem Definition 1

1.2 Contributions 2

1.3 The Outline of the Thesis 3

2 BACKGROUND AND RELATED WORK 5

2.1 Mesh Simplification 5

2.2 Laplace-Beltrami Operator 6

2.2.1 Laplacian Matrix 7

2.2.2 Mass Matrix 9

2.2.3 Spectrum of the Laplace-Beltrami Operator 10
2.3 Related Work 12
3 METHOD 15
3.1 Algorithm 15
3.1.1 Partitioning 19
3.1.2 Multi-Step Simplification with Hierarchical Partitioning 22
3.1.3 Eliminating Similar Edges 24
3.1.4 Threading 25
3.1.5 Illegal Edge Collapses 27
3.1.6 Implementation Details 27
4 EXPERIMENTS AND RESULTS 29
4.1 Evaluation 29
4.2 Results 33
4.2.1 Number of Partitions 33
4.2.2 Number of Eigenvalues 35
4.2.3 Similar Edge Collapses 36
4.2.4 Threading 38
4.2.5 Comparisons 40
4.2.6 Heat Kernel Signature 49
5 CONCLUSION AND FUTURE WORK 51
REFERENCES 53

LIST OF FIGURES

FIGURES

Figure 1.1 Lion mesh is reduced from 20212 vertices to 2000 vertices (10% of its initial size). For an effective spectrum preservation, the functional map visualizations should be resembling the identity matrix. Since the main focus of QSlim [1] is to preserve the appearance of the mesh, it falls short of preserving the spectral properties. 2

Figure 2.1 Edge collapse and vertex split operations 6

Figure 2.2 Edge collapse and vertex split operations for boundary edge . . . 6

Figure 2.3 Uniform Laplacian matrix for a sample mesh 8

Figure 2.4 The angles used in the cotangent weights [2] 9

Figure 2.5 Barycentric cell area for vertex i [3] 9

Figure 2.6 Visualization of the eigenvectors over the mesh surface 11

Figure 3.1 Bunny mesh with $n = 3485$ divided into 10 partitions. The triangles in the edge-cut region are marked with different coloring for better understanding. 20

Figure 3.2 Simplification of the edge-cut region 21

Figure 3.3 Different sized partitions on cactus mesh 22

Figure 3.4 Multi-step hierarchical partitioning for eagle mesh 23

Figure 3.5 Partitions assigned to batches 26

Figure 4.1 Optimal functional map C, where red and blue elements correspond to values 1 and -1 respectively 31

Figure 4.2 Block diagonal C for a sphere mesh simplified from 650 vertices to 200 vertices 32

Figure 4.3 Simplification times for eagle and bunny meshes with different partition sizes 33

Figure 4.4 Results obtained by employing different number of partitions on eagle mesh 34

Figure 4.5 Simplification times for eagle mesh by preserving different number of eigenvalues 35

Figure 4.6 $\|\cdot\|_L$ and $\|\cdot\|_D$ values obtained on bunny mesh for different number of eigenvalues 36

Figure 4.7 Spectrum preservation and timing results for various values of x and n 37

Figure 4.8 Results obtained by enabled and disabling similar edge collapses 38

Figure 4.9 The number of threads (partitions) running at the same time for various number of partitions 39

Figure 4.10 With each method, bunny mesh is simplified from 3485 to 600 vertices (17% of its initial size). 42

Figure 4.11 The first 100 eigenvalues of the Laplacian operator for the coarsened bunny mesh 42

Figure 4.12 With each method, eagle mesh is simplified from 25727 to 2000 vertices (8% of its initial size). 43

Figure 4.13 The first 100 eigenvalues of the Laplacian operator for the coarsened eagle mesh 43

Figure 4.14 With each method, cactus mesh is simplified from 25131 to 2000 vertices (8% of its initial size). 44

Figure 4.15 The first 100 eigenvalues of the Laplacian operator for the coarsened cactus mesh 44

Figure 4.16 With each method, lion mesh is simplified from 20212 to 2000 vertices (10% of its initial size). 45

Figure 4.17 The first 100 eigenvalues of the Laplacian operator for the coarsened lion mesh 45

Figure 4.18 With each method, dragon mesh is simplified from 20741 to 1500 vertices (7% of its initial size). 46

Figure 4.19 The first 100 eigenvalues of the Laplacian operator for the coarsened dragon mesh 46

Figure 4.20 With each method, camel mesh is simplified from 9757 to 800 vertices (8% of its initial size). 47

Figure 4.21 The first 100 eigenvalues of the Laplacian operator for the coarsened camel mesh 47

Figure 4.22 With each method, armadillo mesh is simplified from 49990 to 3000 vertices (6% of its initial size). 48

Figure 4.23 The first 100 eigenvalues of the Laplacian operator for the coarsened armadillo mesh 48

Figure 4.24 Heat kernel signature for the original meshes (left) and their simplified versions (right) 49

LIST OF TABLES

TABLES

Table 4.1 Timings obtained by enabling and disabling threading 38

LIST OF ABBREVIATIONS

3D	3 Dimensional
EVD	Eigenvalue Descriptor
GPS	Global Point Signature
HKS	Heat Kernel Signature

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

Triangular meshes are frequently used to represent 3D models in geometry processing and computer graphics areas. Most of the applications in these areas prefer high resolution models containing tremendous amount of details, in order to provide a more realistic experience. However, as the complexity of a mesh increases, the computational cost required to process it also increases. This is where mesh simplification methods come into the picture, in order to create a simpler version of the complex mesh containing fewer details, by reducing the number of vertices , edges and faces existing in the mesh.

Majority of these mesh simplification methods focus on preserving the appearance of the mesh, which is the case preferred in areas such as rendering. Unfortunately, appearance-preserving methods fall short of maintaining the spectral properties of differential operators constructed on a mesh (see Figure 1.1), which is essential for some geometry processing tasks such as shape correspondence and spectral distance computations. When the simplification is performed by ignoring the spectral properties, the related computations carried out on the coarsened mesh might be inaccurate.

This thesis focuses on the problem of simplifying a 3D triangular mesh, while preserving the spectral properties of the associated Laplace-Beltrami operator. In the recent years, there have been major advancements in spectrum-preserving coarsening methods. However, most of these methods address the problem from a complete algebraic perspective. They rely on directly operating on the matrices corresponding to the differential operators, thus not producing a mesh as output. The only spectrum-

preserving simplification method producing an output mesh is based on utilizing the eigenvectors of the differential operator, maintaining the eigenvalues indirectly. Unlike the previous methods, the method proposed in this thesis considers the eigenvalues of the Laplace-Beltrami operator instead of the eigenvectors, while outputting a simplified triangular mesh. The method is evaluated on a variety of meshes, and it is demonstrated that it preserves the spectrum of the Laplace-Beltrami operator at least as effectively as the existing methods.

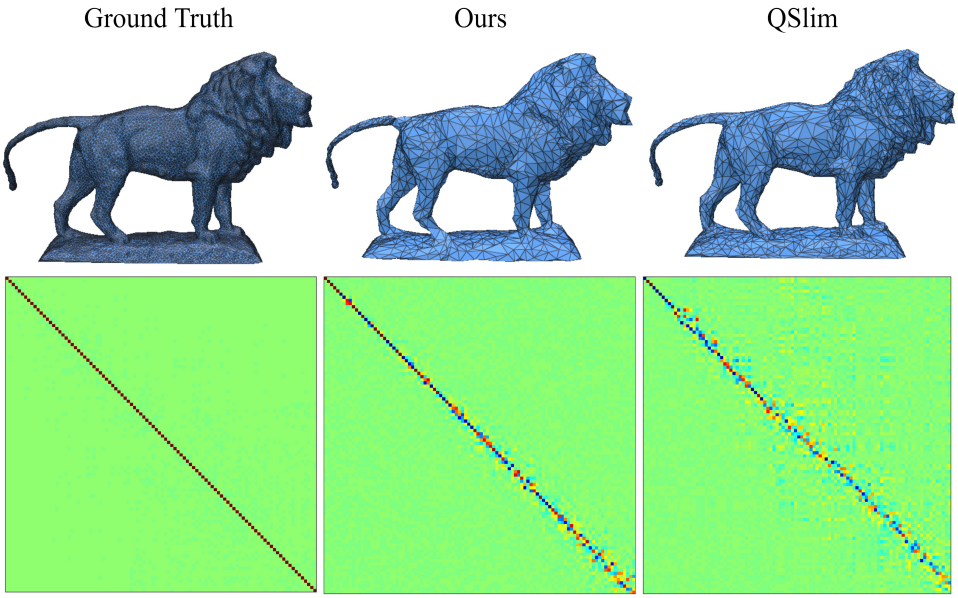


Figure 1.1: Lion mesh is reduced from 20212 vertices to 2000 vertices (10% of its initial size). For an effective spectrum preservation, the functional map visualizations should be resembling the identity matrix. Since the main focus of QSlim [1] is to preserve the appearance of the mesh, it falls short of preserving the spectral properties.

1.2 Contributions

The contributions of the work conducted in this thesis are listed as the following:

- In this thesis, a new partition based mesh simplification method is proposed, whose primary purpose is to preserve the spectrum of the Laplace-Beltrami operator derived from a mesh. The method is capable of preserving the spectrum by considering only the eigenvalues of the operator, whereas the previous

methods are built around the eigenvectors.

- The proposed method is able to perform the spectrum preservation by considering only a small number of eigenvalues, since it is a partition based method.
- Among the existing spectrum-preserving coarsening methods, this method is one of the two methods which produce a mesh as an output.

1.3 The Outline of the Thesis

Chapter 2 introduces the background information related to the method proposed in this thesis. It also includes the review of the existing methods for the related topics in the literature. In Chapter 3, the partition based mesh simplification method proposed in this thesis is explained comprehensively. The spectrum preservation results obtained by employing the method are presented in Chapter 4, along with detailed discussions revolving around the tunable parameters existing in the method. In addition, the comparisons with a variety of previous spectral coarsening methods are included in this chapter. Finally, Chapter 5 provides a summary of the previous chapters, along with possible limitations of the proposed method, which may lead to advancements in the area as future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, firstly, the background information required for understanding the thesis clearly is introduced. How mesh simplification is performed, how the Laplace-Beltrami operator is constructed in the scope of this work and its spectrum are explained in detail. Then, the previous work performed on the related topics are presented.

2.1 Mesh Simplification

Mesh simplification stands for the process of reducing the number of vertices, faces and edges used to represent a mesh, while preserving specific properties of the original mesh as much as possible. These properties often include geometric distances or visual appearance [3]. For mesh simplification, several different operators have been used such as vertex decimation [4], vertex clustering [5] or edge collapse [6]. In this study, edge collapse operator is utilized to simplify the mesh models, as shown in Figure 2.1.

Given an edge e joining two adjacent vertices u and v, an edge collapse operation removes the edge e and replaces the vertices u and v with a new vertex w, by merging them. This operation effectively removes the triangles adjacent to edge e along with e itself, since they become degenerate with the removal of e. Edge collapse operation removes one vertex from the mesh at a time, thus a sequence of successive edge collapse operations are applied to obtain a coarser mesh.

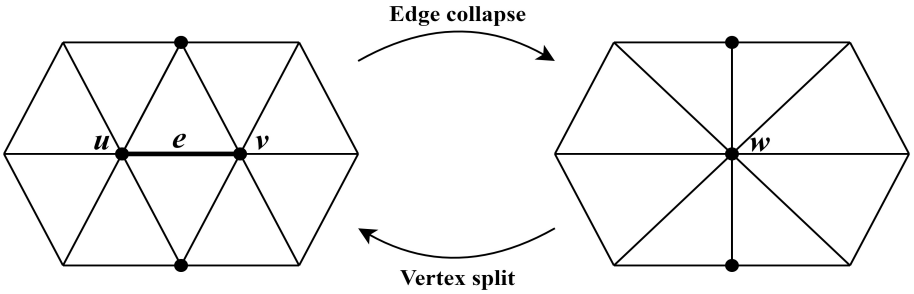


Figure 2.1: Edge collapse and vertex split operations

The inverse of the edge collapse operator is called vertex split [7]. Vertex split operator inserts a new vertex, a new edge and two new triangles to the mesh, as shown in Figure 2.1. In the scope of this thesis, vertex split operator is used only to invert an edge collapse, while in other applications, it is commonly used for increasing the resolution of the surface mesh.

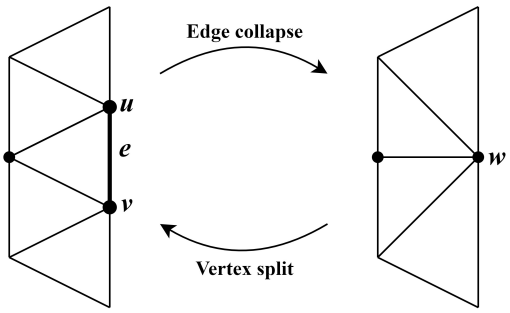


Figure 2.2: Edge collapse and vertex split operations for boundary edge

Figure 2.2 illustrates the edge collapse and vertex split operations for a boundary edge. If edge e is a boundary edge, an edge collapse operation removes only one triangle from the mesh, and vertex split operation introduces only one new triangle to the mesh.

2.2 Laplace-Beltrami Operator

In this section, the matrices forming the Laplace-Beltrami operator, how they are constructed in the scope of this work, and several spectral properties of the Laplace-

Beltrami operator are explained comprehensively. For simplicity, the Laplace-Beltrami operator will be called as the Laplacian operator for the rest of the thesis.

2.2.1 Laplacian Matrix

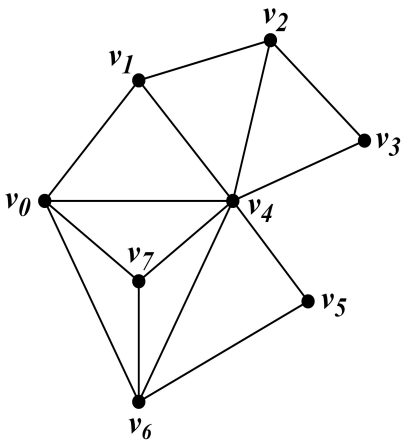
There are several different types of Laplacian matrices that are commonly used in spectral mesh processing, such as the uniform Laplacian (first introduced in [8]), cotangent Laplacian (first proposed in [9]) etc. Uniform Laplacian matrix is build upon only the connectivity of the mesh, whereas the cotangent Laplacian encodes the geometric information of the mesh along with the connectivity. Therefore, the cotangent Laplacian form is preferred in this work.

Let us first define the uniform Laplacian matrix. Given a triangular mesh $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$ with n vertices, $\mathcal{V}$ denotes the set of vertices, $\mathcal{E}$ denotes the set of edges and $\mathcal{F}$ denotes the set of triangular faces. The uniform Laplacian matrix constructed for $\mathcal{M}$ is an $n \times n$ sparse matrix and it can be defined as the following:

$$L_{ij} = \begin{cases} -1 & \text{if } (i,j) \in E \\ d_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

where L_{ij} stands for the element residing on the i^{th} row and the j^{th} column of the matrix L and d_i is the degree of vertex i.

For better understanding, a small sample mesh and its corresponding uniform Laplacian matrix are provided in Figure 2.3. It can be seen that the uniform Laplacian only stores the adjacency information of the vertices for a mesh.



$$L = \begin{bmatrix} 4 & -1 & 0 & 0 & -1 & 0 & -1 & -1 \\ -1 & 3 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 3 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & 7 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ -1 & 0 & 0 & 0 & -1 & -1 & 4 & -1 \\ -1 & 0 & 0 & 0 & -1 & 0 & -1 & 3 \end{bmatrix}$$

Figure 2.3: Uniform Laplacian matrix for a sample mesh

The cotangent Laplacian matrix is obtained by introducing the cotangent weighting scheme on top of the uniform Laplacian. The cotangent Laplacian for $\mathcal{M}$ can be defined as the following:

$$L_{ij} = \begin{cases} -w_{ij} & \text{if } (i,j) \in E \\ \sum\limits_{k \in N(i)} w_{ik} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

where L_{ij} stands for the element residing on the i^{th} row and the j^{th} column of the matrix L, $N(i)$ denotes the vertices in the one-ring neighborhood of vertex i and

$$w_{ij} = \frac{1}{2}(cot\alpha_{ij} + cot\beta_{ij}) \tag{2.3}$$

The angles used in the cotangent weighting scheme are shown in Figure 2.4. It should be noted that for a boundary edge, only one angle is considered, the angle β_{ij} is assumed as zero.

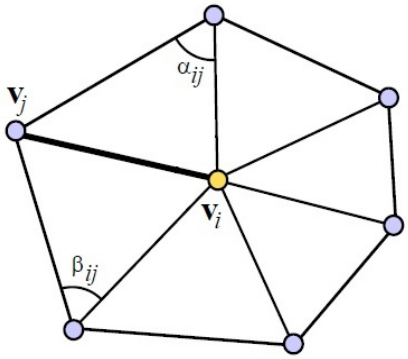


Figure 2.4: The angles used in the cotangent weights [2]

2.2.2 Mass Matrix

Given a triangular mesh $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$ with n vertices, $\mathcal{V}$ denotes the set of vertices, $\mathcal{E}$ denotes the set of edges and $\mathcal{F}$ denotes the set of triangular faces. Mass matrix, also known as the area matrix, for $\mathcal{M}$ is a diagonal matrix with size $n \times n$. For each vertex, it consists of an area measure defined around the one-ring neighborhood of the vertex. This area can be calculated with several different methods such as barycentric cell area, Voronoi cell area or mixed Voronoi cell area [3]. In this thesis, barycentric cell area, shown in Figure 2.5, is preferred, which corresponds to approximately one third of the areas of the one-ring triangles of a vertex as described in Equation 2.4.

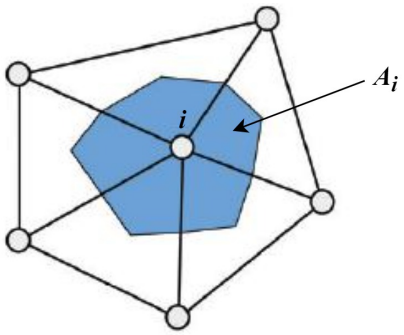


Figure 2.5: Barycentric cell area for vertex i [3]

$$A_i = \frac{1}{3} \sum_{T_j \in N(i)} area(T_j) \tag{2.4}$$

In Equation 2.4, A_i represents the barycentric cell area for vertex i and $T_j \in N(i)$ is the list of triangles lying in the one-ring neighborhood of vertex i. Then, the diagonal mass matrix for the given mesh can be defined as the following:

$$M = \begin{bmatrix} A_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A_n \end{bmatrix} \tag{2.5}$$

2.2.3 Spectrum of the Laplace-Beltrami Operator

Having described the Laplacian matrix L and the mass matrix M, the discrete Laplace-Beltrami operator is defined as in Equation 2.6.

$$\Delta f = M^{-1}L \tag{2.6}$$

Laplacian matrix is a symmetric and positive semi-definite matrix, which leads to real-valued eigenvalues, whose corresponding eigenvectors form an orthogonal basis. Let the eigenvalues of the Laplacian operator be $0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ and their associated eigenvectors be $v_1, v_2, \ldots, v_n$. The spectrum of the Laplacian operator is defined as the set of eigenvalues $\lambda(L) = \{\lambda_1, \lambda_2, ..., \lambda_n\}$ [10]. Therefore, preserving the spectrum of the Laplacian operator of a mesh means preserving the set of eigenvalues as much as possible, which is the main goal of the work described in this thesis.

The standard eigenvalue problem for the Laplacian operator can be written as in Equation 2.7.

$$(M^{-1}L)\vec{v} = \lambda\vec{v} \tag{2.7}$$

Although both L and M are symmetric matrices, the Laplacian operator might be non-symmetric. This results with some problems both from the theoretical perspective and the numerical perspective, since the eigenvalues and eigenvectors of non-symmetric matrices are not guaranteed to be real valued. Therefore, in order to find the eigenvalues and eigenvectors of the Laplace-Beltrami operator, the generalized eigenvalue problem is considered [11]. The general definition of the generalized eigenvalue problem is that given two symmetric matrices $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times n}$, they satisfy the equality given in Equation 2.8.

$$A\vec{v} = \lambda B\vec{v} \tag{2.8}$$

Thus, the generalized eigenvalue problem for the Laplacian matrix $L^{n\times n}$ and the mass matrix $M^{n\times n}$ can be written as in Equation 2.9.

$$L\vec{v} = \lambda M\vec{v} \tag{2.9}$$

This formulation guarantees that all of the generalized eigenvalues and eigenvectors are real-valued. In addition, with L and M being symmetric and positive semi-definite, the generalized eigenvectors are orthogonal to each other.

If the eigen-decomposition of the Laplacian operator is examined from a signal-processing perspective, the Laplacian operator can be interpreted as a discrete signal defined over the mesh surface representing the mesh geometry. Thus, it can be seen that it is analogous to Fourier analysis in a way that the eigenvalues of the Laplacian correspond to the natural frequencies, while the eigenvectors correspond to the natural vibration modes or the harmonics of a surface mesh [8]. This is actually one of the main reasons why the Laplacian operators are exploited in spectral mesh processing tasks frequently.

The eigenvectors can be visualized over the mesh surface as in Figure 2.6, such that the i^{th} entry of an eigenvector corresponds to the magnitude of the eigenvector at vertex v_i, where ϕ_j stands for the eigenvector associated with the j^{th} smallest eigenvalue.

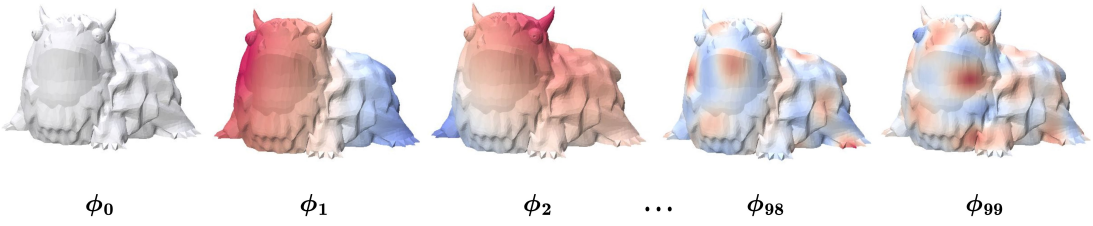


Figure 2.6: Visualization of the eigenvectors over the mesh surface

When the eigenvectors are visualized over the mesh surface, it can be observed that the first few eigenvectors associated with small (low-frequency) eigenvalues look smoother over the surface, compared to the eigenvectors lying at the end of the spectrum associated with large (high-frequency) eigenvalues. This is because the first eigenvectors change slowly over the mesh, unlike the last eigenvectors which are showing rapid changes over the surface. Furthermore, since the value of the first

eigenvalue is zero, the corresponding eigenvector ϕ_0 is a constant vector, which can be interpreted as the lowest frequency or the smoothest function [2].

If the elements of the matrix corresponding to the discrete Laplacian operator are examined separately, it is observed that they only stand for some local information gathered around the local neighborhood of a vertex. However, when the eigenvalues and the eigenvectors of the Laplacian operator are examined, they turn out to be capable of storing the global characteristic information about the mesh. One simple example can be that the multiplicity of the zero eigenvalue corresponds to the number of connected components in the mesh [10].

2.3 Related Work

This section addresses the previous work related to mesh simplification, the use of Laplace-Beltrami operator in spectral geometry processing and spectral coarsening, considering the topics that the method presented in this thesis is built on.

Mesh simplification has been a well-studied topic in computer graphics area due to the growing need to be able to represent the meshes at different resolutions corresponding to different level-of-details. For the current context, mesh simplification methods can be categorized as appearance-preserving and spectrum-preserving methods. The majority of the simplification methods are focused on preserving the appearance and the geometric properties of the mesh. The most prominent previous examples in this area include mesh optimization [6] and mesh decimation [4]. One of the most well-known algorithms among the appearance-preserving simplification methods is the quadric error metrics method introduced by Garland and Heckbert [1]. Later, this greedy edge collapse algorithm is extended to preserve a variety of vertex attributes such as textures, colors or normal vectors along with the geometry [12, 13]. For the appearance-preserving methods, [14] provides a comprehensive study, where multiple mesh simplification methods are examined and compared. Even though these methods manage to preserve the appearance of the mesh successfully, they fall short of maintaining the spectral properties of differential operators constructed on a mesh, which lie at the core of some geometry processing tasks such as shape correspon-

dence.

Laplace-Beltrami operator has been utilized for a variety of spectral geometry processing tasks for many years. The use of Laplacian operator for mesh processing was first introduced by Taubin [8], pointing out the analogy between the Laplacian operator and the Fourier analysis. The eigenvalues and eigenvectors of the Laplacian are then exploited in areas such as mesh segmentation [15,16], shape correspondence [17] and mesh compression [18]. Karni and Gotsman [18] performed spectral mesh compression by dividing the mesh into smaller patches to reduce the computational cost, similar to the partitioning carried out in our method.

Alongside these studies, since the Laplacian operator is invariant under isometric transformations, it also served as a robust foundation for deformation-invariant shape descriptors. With this purpose, the eigenvalues of the Laplacian operator are utilized for extracting fingerprints called shape-DNA, to represent a surface or a mesh [19,20]. These works have proved that the spectrum of the Laplacian has a discriminative power that is capable of capturing the global properties of a shape. However, it should also be noted that the spectrum of the Laplacian does not provide a complete identification for the shape, since there are some rare cases, where two non-isometric shapes have the same spectrum. The use of eigenvalues alone as shape descriptors is one of our source of inspirations for depending this simplification method on the eigenvalues of the Laplacian operator. Following these works, shape signatures called global point signature (GPS) [11] and heat kernel signature (HKS) [21] are introduced, including the eigenvectors into the scenario as well. These shape descriptors are utilized in works such as detecting global intrinsic symmetries of the shapes by Ovsjanikov et al. [22].

For more information about the work conducted on mesh processing using the Laplacian framework and the use of spectral methods in various fields of geometry processing, it is recommended that the surveys presented by Sorkine [2] and Zhang et al. [10] are examined.

In the recent years, there has been significant developments in simplification methods which are focused on preserving the spectral properties of a mesh rather than just the appearance. Öztireli et al. [23] resampled points on a manifold surface by pre-

serving the spectrum of the Laplacian operator. Similarly, Liu et al. [24] presented a spectrum-preserving coarsening method, which is also built on sampling points from the original mesh. Their method can be directly applied to the discrete geometric operators derived from a mesh including the Laplacian operator. They also introduced a metric build upon functional maps [25] to measure how well the spectrum of an operator is preserved after the coarsening. The mentioned metric is utilized in this work, as well as in [26] and [27]. However, both of the proposed methods do not produce a mesh. Later, Lescoat et al. [26] proposed a spectral mesh simplification method built upon the formulation presented in [24], which produces a mesh as output. They altered the greedy edge collapse algorithm introduced by Garland and Heckbert [1] with a spectral cost metric. Their spectral cost relies on the eigenvectors of the Laplacian while preserving the spectrum, whereas the method proposed in this thesis focuses directly on preserving the eigenvalues of the Laplacian operator. Following the work of Liu et al. [24], Chen et al. [27] proposed an operator coarsening scheme using chordal decomposition, enabling the optimization of an operator separately from the mesh. The main difference that separates our method from the existing spectral simplification methods is that our method solely relies on the eigenvalues of the Laplacian operator, whereas the others utilize the eigenvectors.

CHAPTER 3

METHOD

The method proposed in this thesis is a spectrum-preserving mesh simplification algorithm based on edge collapse operations. The main goal of the method is to preserve the spectral properties of the input mesh at the low frequencies as much as possible, while reducing the number of elements used to represent the mesh. Preserving the spectral properties of the mesh indicates maintaining the spectrum of the Laplacian operator defined on the mesh, which corresponds to the eigenvalues of the Laplacian operator as stated in Section 2.2.3. However, it is a known fact that the eigen-decomposition of large matrices like the Laplacian operator defined on a high-resolution mesh is a very costly computation. Thus, instead of preserving the whole spectrum, the proposed method only focuses on preserving the low frequency end of the spectrum. The reason for that is the high-frequency components will not be present on the coarser domain [24]. Therefore, there is no reason for trying to preserve the high frequency end of the spectrum. This situation allows to focus on the more meaningful part of the spectrum and instead of computing the whole spectrum of the operator, the eigen-decomposition can be performed for only the first k components.

In the upcoming sections in this chapter, the details of the work conducted, how the spectrum-preserving simplification method works, the algorithm and the related parameters are explained.

3.1 Algorithm

The algorithms used in this work are presented in Algorithm 1 and Algorithm 2.

Algorithm 1 Spectrum-Preserving Simplification

Input: $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F}), m, k, p, n, x$

Output: $\widetilde{\mathcal{M}} = (\widetilde{\mathcal{V}}, \widetilde{\mathcal{E}}, \widetilde{\mathcal{F}})$

$\widetilde{\mathcal{V}} \leftarrow \mathcal{V}; \widetilde{\mathcal{E}} \leftarrow \mathcal{E}; \widetilde{\mathcal{F}} \leftarrow \mathcal{F}$

Divide the mesh $\widetilde{\mathcal{M}}$ into smaller partitions $\widetilde{\mathcal{M}}_1, \widetilde{\mathcal{M}}_2, \ldots, \widetilde{\mathcal{M}}_p$ where $\widetilde{\mathcal{M}}_i = (\widetilde{\mathcal{V}}_i, \widetilde{\mathcal{E}}_i, \widetilde{\mathcal{F}}_i)$

Assign partitions $\widetilde{\mathcal{M}}_1, \widetilde{\mathcal{M}}_2, \ldots, \widetilde{\mathcal{M}}_p$ to threads $T_1, T_2, \ldots, T_p$

Calculate the number of edges that will be collapsed in each partition $nE_1, nE_2, \ldots, nE_p$

Assign threads $T_1, T_2, \ldots, T_p$ to thread batches $B_1, B_2, \ldots, B_b$

for batch B_i in $B_1, B_2, \ldots, B_b$ **do**

  for thread T_j in batch B_i **do**

    $\Lambda_{in} = \{\lambda_{in_1}, \ldots, \lambda_{in_k}\} \leftarrow$ the first k eigenvalues of the Laplacian for partition $\widetilde{\mathcal{M}}_j$

    while $nE_j > 0$ **do**

      $allCosts \leftarrow FindEdgeCosts(\widetilde{\mathcal{M}}_j, \widetilde{\mathcal{E}}_j, k, \Lambda_{in})$

      $bestEdges \leftarrow$ edgeIds of $allCosts[0:n]$

      Collapse $bestEdges[0]$; $nE_j \leftarrow nE_j - 1$

      for i in $1, \ldots, x$ **do**

        $costs \leftarrow FindEdgeCosts(\widetilde{\mathcal{M}}_j, bestEdges, k, \Lambda_{in})$

        Collapse $costs[0]$; $nE_j \leftarrow nE_j - 1$

      end for

    end while

  end for

end for

Algorithm 2 FindEdgeCosts

Input: $\widetilde{\mathcal{M}}_j = (\widetilde{\mathcal{V}}_j, \widetilde{\mathcal{E}}_j, \widetilde{\mathcal{F}}_j), \mathcal{E}, k, \Lambda_{in}$

Output: $costs$

$costs \leftarrow \{\}$
for edge $e \in \mathcal{E}$ **do**
  $result \leftarrow$ Collapse edge e
  if $result$ is $successful$ **then**
    $\Lambda_{out} = \{\lambda_{out_1}, \ldots, \lambda_{out_k}\} \leftarrow$ the first k eigenvalues of the Laplacian for partition $\widetilde{\mathcal{M}}_j$
    $cost_e \leftarrow$ the difference between Λ_{in} and Λ_{out} wrt L_{evd} norm
    Reverse the edge collapse
  end if
  Add $(e, cost_e)$ into $costs$
end for
Sort $costs$ wrt increasing cost
return $costs$

The input to the algorithm is a manifold triangular mesh $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$, which can possibly contain boundaries. After the simplification process, it outputs a coarser mesh $\widetilde{\mathcal{M}} = (\widetilde{\mathcal{V}}, \widetilde{\mathcal{E}}, \widetilde{\mathcal{F}})$ with spectral properties as close as possible to $\mathcal{M}$. In addition, the algorithm also requires that the following inputs are provided:

- m: the desired number of vertices in the simplified mesh
- k: the number of eigenvalues to preserve
- p: the number of partitions which the mesh will be divided into
- x: the number of similar edge collapses that will take place, after choosing the best edge to collapse for the current step by considering all the edges in a partition
- n: the number of edges to consider while choosing the similar edge collapses

The main idea lying at the core of the algorithm is to compare the distance between the eigenvalues of the Laplacian computed on the original mesh and the eigenvalues after

an edge collapse, for measuring spectrum preservation. Since the method is focused on preserving the low frequency components of the spectrum, instead of calculating and comparing all the eigenvalues of the operator, only the smallest k eigenvalues are considered. The parameter k provided to Algorithm 1 specifies the number of eigenvalues aimed to be preserved. For both the shape descriptors described in [11] and the global surface representation performed in [28], the number of eigenvalues found to be sufficient for representing the global properties of a surface changes from 20 to 25. Based on these sources of inspiration, k is selected as 15 for the presented method, unless otherwise stated.

Algorithm 2 summarizes the main eigenvalue comparison method mentioned. It is based on collapsing the edges given in the edge set $\mathcal{E}$ one by one and associating each edge with a cost measuring how much the spectrum of the Laplacian operator is affected from the current collapse. After each edge is assigned a cost, they are sorted with respect to increasing cost. The cost associated with each edge is calculated by comparing the set of first k eigenvalues with respect to a norm. During the implementation process, several different norms have been tried including the $L2$ norm, but L_{evd} norm has outperformed the others. L_{evd} norm is originated from the work conducted in [28]. In their work, the eigenvalues of the affinity matrix operator is used as a shape descriptor, called the eigenvalue descriptor (EVD). In order to measure the distance between two eigenvalue descriptors, they utilize the norm provided in Equation 3.1. Here, $\lambda_i^{\mathcal{M}}$ and $\lambda_i^{\widetilde{\mathcal{M}}}$ stand for the i^{th} eigenvalues of the Laplacian operators derived from the input and output meshes respectively.

$$L_{\text{evd}}(\mathcal{M}, \widetilde{\mathcal{M}}) = \frac{1}{2} \sum_{i=1}^{k} \frac{\left[|\lambda_i^{\mathcal{M}}|^{\frac{1}{2}} - |\lambda_i^{\widetilde{\mathcal{M}}}|^{\frac{1}{2}} \right]^2}{|\lambda_i^{\mathcal{M}}|^{\frac{1}{2}} + |\lambda_i^{\widetilde{\mathcal{M}}}|^{\frac{1}{2}}} \tag{3.1}$$

In order to make sure that the selected edge collapse would result with the best preservation of the spectrum amongst the current choices, all of the edges are considered one by one. However, this approach requires the eigenvalues of the Laplacian to be computed after each collapse for all the edges in the mesh, at each step. As might be expected, this results with a highly infeasible amount of time to wait even for selecting just a single edge to collapse. To give an example, for a mesh with $n = 50000$ vertices, the Laplacian matrix has size 50000×50000. The computation of the first 15 eigenvalues for this Laplacian takes approximately 1.4 seconds in our environ-

ment. This means that in order to choose a single edge which preserves the spectrum the most, this eigen-decomposition process should be applied for each edge in the mesh. To overcome this impractical situation, a variety of approaches are introduced to the algorithm, which are explained in detail throughout this section, to make it as efficient as possible. Even though the algorithm is built around this basic comparison idea, with the methods introduced on top of it, critical timing improvements are achieved, without sacrificing quality, as will be shown in Section 4.2.

3.1.1 Partitioning

The first approach introduced on top of the core idea is partitioning. The main algorithm presented in Algorithm 1 starts by dividing the mesh into smaller partitions, in order to decrease the amount of suffering from high computational cost of eigenvalue decomposition. Each partition is simplified separately, thus the Laplacian matrices are constructed within the partition, instead of constructing them for the whole mesh. This effectively decreases the size of the Laplacian matrix, leading to much faster eigenvalue computations. In addition to that, when each partition is treated separately, there are fewer edges to consider while choosing the best edge for a single collapse. Apart from timing concerns, partitioning also has the advantage to capture the properties of the local neighborhoods better. A sample partitioning is provided in Figure 3.1.

Alongside its benefits, partitioning the vertices may introduce some problems to the visual quality of the output mesh. When each vertex is assigned to a partition, the triangles left between the partition boundaries constitute the edge-cut regions, as can be seen in Figure 3.1. To minimize the possible negative outcomes of this, the partitioning strategy used should prioritize minimizing the edge-cut regions as much as possible, while keeping the partitions well-balanced, in a way that the number of vertices assigned to each partition are close to each other. However, it should be noted that finding the optimal partitioning of a mesh while minimizing the number of edges lying inside the edge-cut region is an NP-complete problem [29]. For these reasons, a partitioning tool called METIS is utilized in this work, which provides an option to prioritize minimizing the edge-cut regions to an extent that is sufficient for our pur-

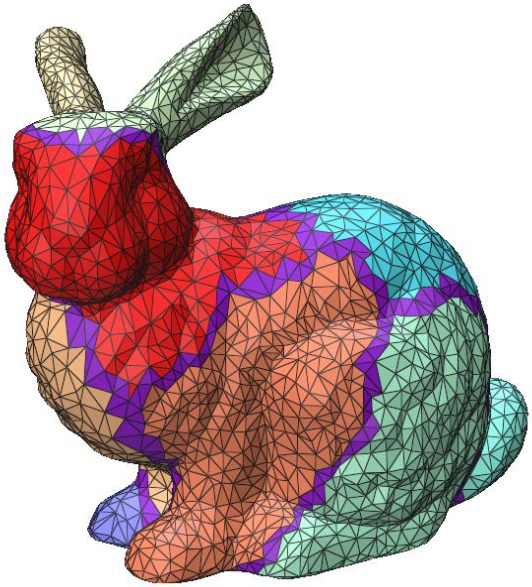


Figure 3.1: Bunny mesh with $n = 3485$ divided into 10 partitions. The triangles in the edge-cut region are marked with different coloring for better understanding.

poses [30]. This partitioning algorithm has also been used by Karni and Gotsman [18] for partitioning the mesh into smaller sub-meshes with the same purpose described here, for spectral compression tasks.

The partitioning used in this work guarantees that all edges belong to only one partition, in order to prevent the partitions from affecting each other. Actually, this constraint is the reason why the edge-cut regions occur in the first place. Since the edges lying inside the edge-cut region do not belong to any partition, they are not collapsed explicitly. It might be expected that this would result with an uneven triangulation on the simplified mesh surface, where the edge-cut region stays the same as it was in the input mesh, while the regions inside the partitions are simplified. Fortunately, that is not the case, because the edge-cut region gets simplified indirectly, as the edges on the partition boundaries are collapsed. This indirect simplification is illustrated in Figure 3.2. The edges highlighted with bold lines represent the partition boundaries, where dark blue triangles belong to partition P_1 and dark green triangles belong to partition P_2. If edge e lying on the boundary of partition P_2 is collapsed, it can be observed that it results with the removal of a triangle inside the edge-cut region. Light blue triangles represent the triangles which will be removed indirectly with the collapse of

an edge lying on the boundary of partition P_1, whereas light green triangles represent the ones which will be removed with the collapse of an edge lying on the boundary of partition P_2. In this way, all of the triangles inside the edge-cut region are also included in the simplification process and uneven triangulation on the coarsened mesh is prevented as much as possible.

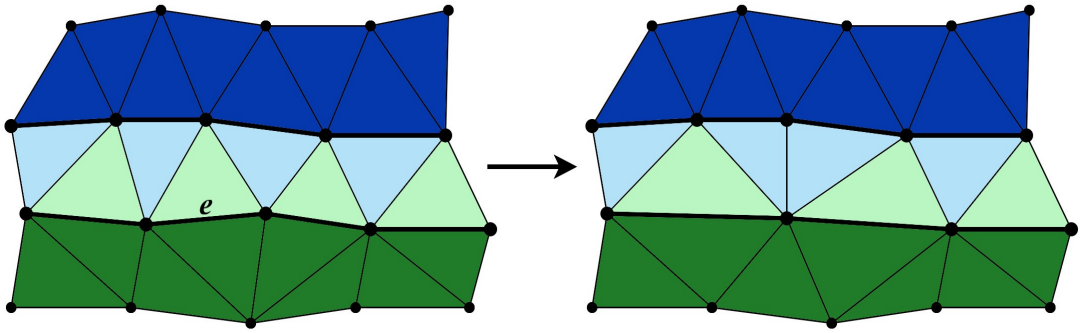


Figure 3.2: Simplification of the edge-cut region

While partitioning the mesh into smaller patches, there are several important points to take into consideration. One of them is choosing the size of the partitions. In order to allow the partitions to represent the global properties of the mesh along with the local properties, the partition sizes should be adjusted carefully. Since each partition is simplified within itself and has its own Laplacian operator built specifically for that partition, they are not aware of the shape, the connectivity or the spectral properties represented in the remainder of the mesh. If the partition sizes are too small, they are not capable of representing the global properties enough. Figure 3.3 shows an example for this situation, where the partition shown on the right is a better selection than the left one, because the patch selected on the left is insufficient to represent its overall neighborhood. On the contrary, if the partition sizes are too large, the timing concerns jump back into the scene. Consequently, a sweet spot should be found for the size of the partitions, to achieve the balance between providing noticeable timing improvements and managing to maintain the global properties adequately. Partition sizes are determined by the input parameter p in Algorithm 1, which implies the number of partitions. In Section 4.2.1, the results obtained with several different partition sizes are presented.

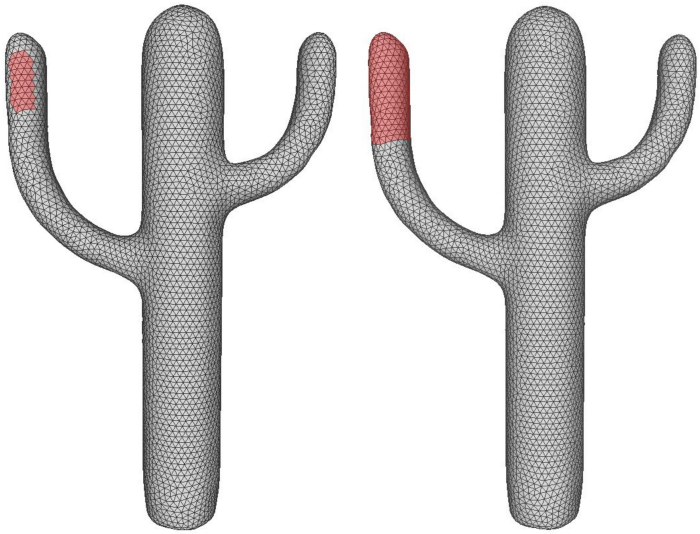


Figure 3.3: Different sized partitions on cactus mesh

Another important point is to decide how many edges should be collapsed from each partition. A number of approaches can be considered for this. However, in this work, same number of edges are collapsed from each partition, since the partitions are balanced.

As experimentally demonstrated for the simplification works performed in [24] and [26], the number of vertices left in the coarsened mesh should be at least 3 times the number of eigenvalues to preserve, in order to preserve the spectrum of the Laplacian operator properly. Since the Laplacian operators are constructed individually for each partition, the number of vertices that should be present in a partition on the simplified mesh should be at least $3 \times k$. Hence, this situation bounds the number of vertices that can be removed from a partition, allowing the simplification only up to a point, for correct preservation of the spectrum. To overcome this downfall of partitioning, multi-step simplification with hierarchical partitioning is introduced to the method.

3.1.2 Multi-Step Simplification with Hierarchical Partitioning

If the desired output size m could not be achieved with the given number of partitions p, hierarchical partitioning is applied for multiple levels until m could be achieved.

By employing multi-step simplification with hierarchical partitioning, the simplification process is performed in multiple steps. At each step, the mesh is simplified until an intermediate resolution, which satisfies the constraint mentioned above. The number of simplification steps, desired resolutions and the number of partitions at the intermediate steps are determined according to the input parameters p and m.

Assume that for eagle mesh, the p parameter was supplied as 100, where the desired resolution m is much lower than the resolution that can be achieved with 100 partitions. In this case, the intermediate resolutions and the number of partitions are calculated in a way that m could be achieved. Let the calculated number of partitions for the intermediate steps be 100, 50 and 25. The hierarchical partitioning of the mesh for this case can be illustrated as in Figure 3.4.

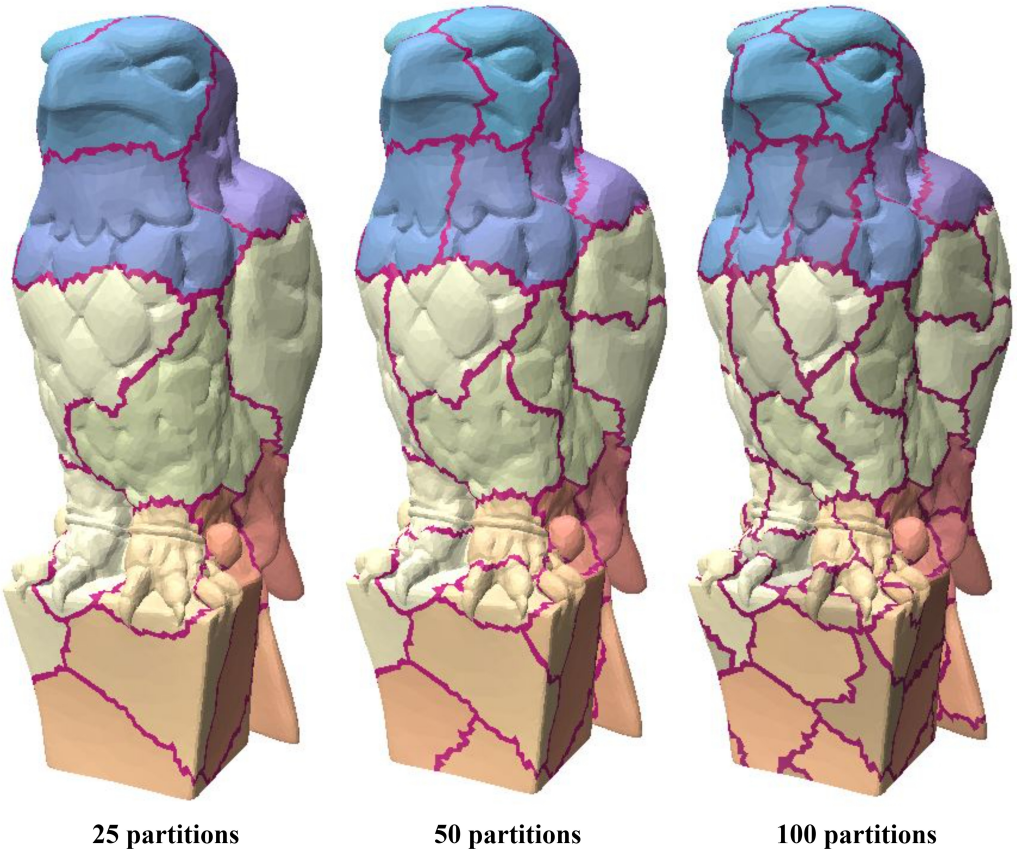


Figure 3.4: Multi-step hierarchical partitioning for eagle mesh

In order to obtain the described hierarchical partitioning, firstly, the mesh is divided into 25 partitions. Then, each of these 25 partitions are divided into 2 to obtain 50

partitions. Since m is still could not be reached, the hierarchical partitioning is applied for one more step. For this, each of the resulting 50 partitions are divided into 2 smaller partitions to obtain 100 partitions. Here, it can be observed that hierarchical partitioning is actually analogous to a tree structure, as each partition is divided into smaller patches step by step. Having applied the hierarchical partitioning, the simplification is performed from the level containing more partitions to the level containing less partitions. Hence, it starts with the step consisting of 100 partitions, then it continues with 50 and 25 partitions respectively.

3.1.3 Eliminating Similar Edges

In order to speed up the edge selection process a bit more, an approach based on removing a certain number of edges similar to the selected edge is followed. For this purpose, a variety of metrics are considered such as Gaussian curvature, Global Point Signature [11] and Heat Kernel Signature [21], to find out what "similar" means in this context. However, exploiting these metrics has deteriorated the spectrum preservation results slightly, so a simpler metric is preferred.

Firstly, the meaning behind a similar edge collapse should be defined. Assume that edge e is collapsed for the current step. In order to find other edges that are similar to e, we should look for edges that are spectrally close to e. This means that for two edge collapses to be similar to each other, they should be affecting the spectrum with an amount as close as possible to each other.

As explained before, at each step, the costs are calculated for all the edges in the partition according to how much they affect the original spectrum. Then, the edges are ordered with respect to this cost. At this point, instead of just choosing the topmost edge with the minimum cost, top n edges can be extracted. The first thing that comes to mind is to collapse these top n edges successively. However, this results with either a degradation in the spectrum preservation or a deformation in the shape of the output mesh. Thus, rather than collapsing all the extracted edges without any checks, we can select a number of edges to collapse among these top n edges. Let this number be x. To make it more clear, assume that for the first step all possible edge collapses in the partition are reviewed and top n edges with the minimum costs are picked. Let this

edge set be $\mathcal{E} = \{e_1, e_2, \ldots, e_n\}$, where the edges are ordered with respect to increasing cost. For the first step, edge e_1 is collapsed, since currently it is the edge with the minimum cost. For the second step, the best edge is selected among the remaining edges in the set $\mathcal{E}$, which corresponds to $\{e_2, e_3, \ldots, e_n\}$. Assuming that edge e_3 is selected for the second step, for the third step only the edges in $\{e_2, e_4, \ldots, e_n\}$ are considered. This process is repeated for a total of x iterations.

With this approach, all edges in the partition are only reviewed once every x steps instead of at each step. In the remaining steps, eigenvalue calculations are only performed for the edges in the set $\mathcal{E}$. Considering that the size of $\mathcal{E}$ is much smaller than the number of edges in the partition, this results with a substantial amount of acceleration in the algorithm. The x and n values mentioned here are provided to the algorithm as input parameters. It should be noted that n must be greater than $3x$, because some of the n edges will be removed indirectly through other edge collapses in this subset.

3.1.4 Threading

The final acceleration technique introduced to the method is threading. Since the mesh is already divided into a number of partitions and they are all treated separately, the first thing that comes to mind is to run the partitions in parallel by assigning each partition to a thread. However, if all the partitions are simplified concurrently, the neighboring partitions affect each other. Because a vertex lying on the boundary of partition P_1 has part of its neighborhood in partition P_2, which is a direct neighbor of P_1. This results with race conditions between the threads. In order to eliminate the potential race conditions without utilizing concurrency control mechanisms and to avoid making the method more complicated, threads are run batch by batch, in a way that no two neighboring partitions are run at the same time. The problem of assigning the threads into batches is analogous with the well-known graph coloring problem, for which finding the optimal solution is an NP-complete problem [29]. However, using one of the greedy solutions providing an upper bound on the number of batches is sufficient for our purposes.

The greedy solution utilized here orders the vertices with respect to their degrees de-

creasingly and assigns the colors to vertices in that order [31]. In our case, vertices correspond to partitions, where a partition's degree is defined as the number of its neighboring partitions. In addition, colors correspond to batches, in which the threads are assigned. By employing this greedy algorithm, the number of thread batches required is guaranteed to be at most one more than the maximum number of neighbors a partition has. Furthermore, in this way, significant time improvements and deterministic results are achieved through threading, while ensuring that no partition affect each other.

Figure 3.5: Partitions assigned to batches

Figure 3.5 shows an example of how the partitions are assigned to thread batches. The partitions running at the same time are marked with the same color, where each color represents a different thread batch.

3.1.5 Illegal Edge Collapses

While performing an edge collapse, it should be checked whether it is a legal operation or not. To avoid illegal collapses, the following checks are performed before collapsing edge e:

- If both endpoints of the edge are boundary vertices, edge e must be a boundary edge [6]. This ensures preserving boundaries of the mesh.
- All joint neighbors of the endpoints of the edge must be in a face with e [6]. This ensures the mesh is still manifold.
- For each neighboring face of edge e, the normals of the triangles should be compared before and after the collapse as in [1]. If there is a normal flip, the edge collapse should not be allowed to ensure that triangles do not flip, thus avoiding mesh inversion.
- If after the edge collapse, the maximum dihedral angle of the edges in one-ring neighborhood of e is greater than some threshold, the edge collapse should be disallowed [6]. This is a heuristic check ensuring that the mesh does not intersect itself after the edge collapse.

3.1.6 Implementation Details

In this section, several details about the implementation of the proposed method are provided.

- The method is implemented with C++.
- Eigen library [32] is utilized for sparse and dense matrix operations.
- Spectra library [33], which is built on top of Eigen, is used for eigenvalue decomposition of the Laplacian operator.

CHAPTER 4

EXPERIMENTS AND RESULTS

This chapter presents the mesh simplification results obtained with the proposed method for different meshes. The metrics used for evaluation, the results achieved by testing different parameters and comparisons with several previous methods are demonstrated with details. All the results presented are obtained on a machine with an Intel i7-6700HQ 2.60 GHz CPU and 16 GB of RAM.

4.1 Evaluation

In order to measure how well the spectrum of a mesh is preserved after the simplification, functional maps and the quantitative metrics introduced by Liu et al. [24] and Lescoat et al. [26] are used.

Functional maps [25] describe a way to transfer functions from one shape to another. In spectral mesh simplification context, they are utilized as a measure to evaluate how well the eigenvectors of the Laplacian operator $\widetilde{L} \in \mathbb{R}^{m \times m}$ computed on the output mesh $\widetilde{\mathcal{M}}$ represents the eigenvectors of the Laplacian operator $L \in \mathbb{R}^{n \times n}$ computed on the input mesh $\mathcal{M}$. In order to take only the low frequency components into account, the functional map C will be considered for the first k eigenvectors. Therefore, the functional map $C \in \mathbb{R}^{k \times k}$ between the input and output meshes can be defined as the following:

$$C = \widetilde{\Phi}^T \widetilde{M} P \Phi \tag{4.1}$$

In Equation 4.1, $\Phi \in \mathbb{R}^{n \times k}$ and $\widetilde{\Phi} \in \mathbb{R}^{m \times k}$ are the set of eigenvectors corresponding to the first k eigenvalues of the Laplacian operator obtained on the input and output meshes respectively. $\widetilde{M} \in \mathbb{R}^{m \times m}$ stands for the mass matrix constructed on the output

mesh, and $P \in \mathbb{R}^{m \times n}$ represents the projection matrix.

Projection matrix is a fine-to-coarse operator allowing to transfer functions across meshes at different resolutions. Since the Laplacian operators computed on the input and output meshes have different sizes, their corresponding eigenvectors are of different lengths as well. In order to be able to compare them in a meaningful way, projection matrix P is utilized in the functional map. P can be computed during the simplification process as in [26] or it can be computed as a subset selection operator as in [24]. In this work, it is also computed during simplification like Lescoat et al. did [26].

Throughout the simplification process, each edge collapse operation is associated with a restriction matrix Q, summarizing the changes in the vertex set of the mesh caused by the edge collapse operation. $Q \in \mathbb{R}^{n-1 \times n}$ is defined as in Equation 4.2, where n denotes the number of vertices existing in the mesh before the collapse.

$$V^{'} = QV \tag{4.2}$$

In Equation 4.2, V denotes the vertex set before the edge collapse, whereas $V^{'}$ denotes the vertex set after the collapse.

Assume that edge e, joining vertices u and v is collapsed, where u is kept while v is deleted from the mesh. Let $u^{'}$ be the index of u after collapse, also the index of the merged vertex, and $x^{'}$ be the index of vertex x after collapse. Considering this notation, the non-zero elements in the Q matrix are given as the following:

$$\begin{aligned} &Q_{u'u} = 0.5 \\ &Q_{u'v} = 0.5 \\ &\forall x \in V - \{u, v\}, Q_{x'x} = 1 \end{aligned} \tag{4.3}$$

In general, the merged vertex position can be calculated by interpolating between the positions of u and v, as in Equation 4.4. It should be noted that in this work α is selected as 0.5, since merged vertex position is the midpoint of the edge.

$$u^{'} = (1 - \alpha)u + \alpha v \tag{4.4}$$

In order to obtain the projection matrix P, Algorithm 1 described in Chapter 3 should be modified in a way that initially P matrix is set to identity, and with each edge

collapse, it is updated with the Q matrix associated with that collapse. This can be defined as the following:

$$P = Q_n Q_{n-1} \dots Q_2 Q_1 \tag{4.5}$$

where Q_i is the restriction matrix for the i^{th} edge collapse.

Having calculated P, C matrix can also be defined as an inner product matrix between the first k eigenvectors of L and $\widetilde{L}$ [24]. Since the eigenvectors are orthonormal, the optimal functional map C should be a diagonal matrix filled with values -1 and 1 [27], as shown in Figure 4.1.

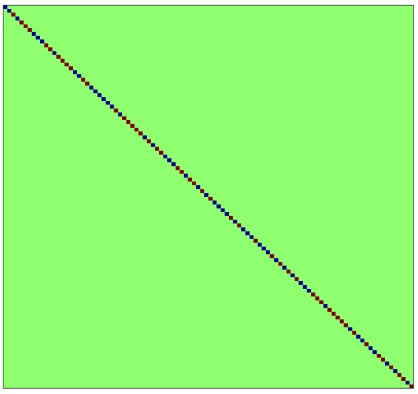


Figure 4.1: Optimal functional map C, where red and blue elements correspond to values 1 and -1 respectively

The closer the C matrix to a diagonal matrix, the better the spectrum is preserved. It should be noted that if the eigenvalues have multiplicity, C will be in block-diagonal form, as in the case for a sphere mesh (see Figure 4.2).

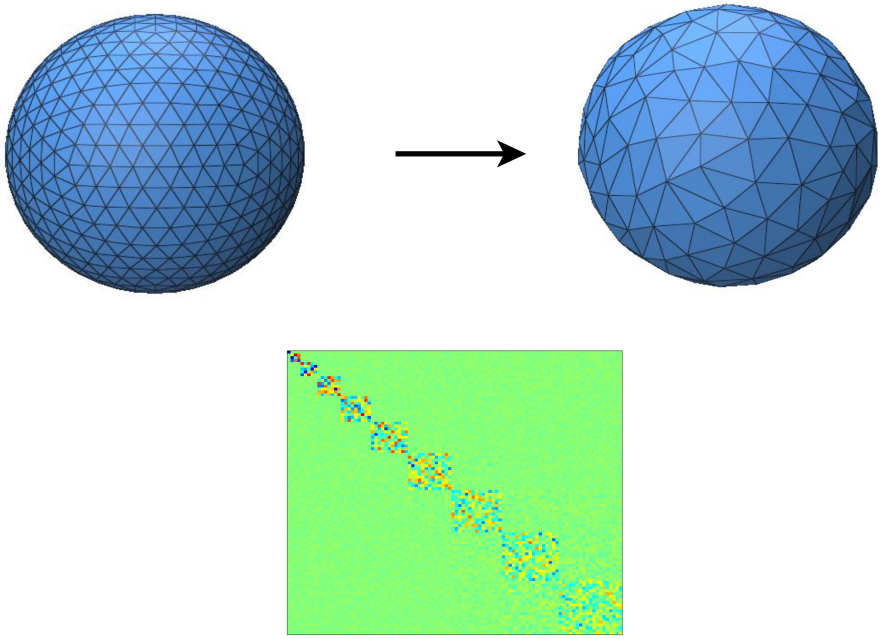


Figure 4.2: Block diagonal C for a sphere mesh simplified from 650 vertices to 200 vertices

For not relying only on the comparisons of the visualizations of the C matrix, the following norms are introduced in [26] as quantitative metrics:

$$\begin{aligned} \text{Laplacian commutativity: } \|C\|_L^2 &= \frac{\|C\Lambda - \widetilde{\Lambda}C\|^2}{\|C\|^2} \\ \text{Orthonormality: } \|C\|_D^2 &= \|C^T C - Id\|^2 \end{aligned} \tag{4.6}$$

where $\Lambda, \widetilde{\Lambda} \in \mathbb{R}^{k \times k}$ are diagonal matrices storing the first k eigenvalues calculated on the input and output meshes respectively. Here, the Laplacian commutativity norm is originated from the idea that the eigenvalues of the Laplacian should not change between the original and the simplified meshes. On the other hand, the orthonormality norm is used as a quantitative measure to tell how close the functional map to the identity matrix. Therefore, for these norms, the closer the values are to zero, the better the spectrum is preserved throughout the simplification.

4.2 Results

For all the results that are presented in this section, the functional maps are of size 100×100. Although the proposed method is only focused on preserving the first 15 eigenvalues, the functional maps are provided for the first 100 eigenvalues to indicate the distribution behind the first 15, as well.

4.2.1 Number of Partitions

As the mesh is divided into more partitions, the size of each partition gets smaller. Consequently, with each partition having fewer vertices, the size of the Laplacian operators constructed for each partition are reduced, leading to faster eigenvalue computations. Besides, since the partitions are run concurrently via threading, as the number of partitions increases, the time it takes to simplify the overall mesh decreases. Figure 4.3 shows the timing results obtained by employing different partition sizes for eagle and bunny meshes. Bunny mesh originally has 3485 vertices and it is simplified until 1000 vertices remain, whereas the eagle mesh originally contains 25727 vertices and it is simplified until 14000 vertices left.

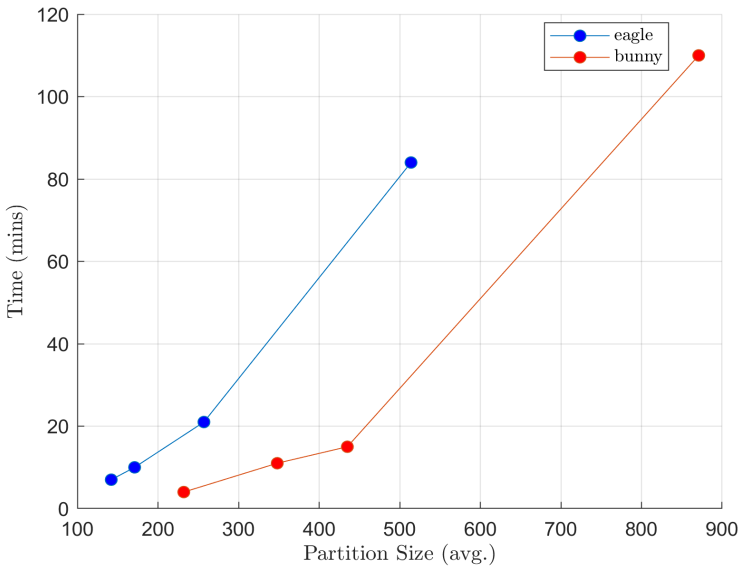


Figure 4.3: Simplification times for eagle and bunny meshes with different partition sizes

In Figure 4.3, partition size stands for the average number of vertices in a partition. It can be observed that as the partition sizes shrink or as the number of partitions increase, the total simplification time decreases. However, the partition sizes should not be selected to be the smallest possible by only considering the timing improvements, as explained in Section 3.1.1, since the main goal of this method is to preserve the spectrum of the mesh. For each mesh, partition sizes are adjusted so that the acceleration amount is sufficient to compensate the loss in the spectrum preservation.

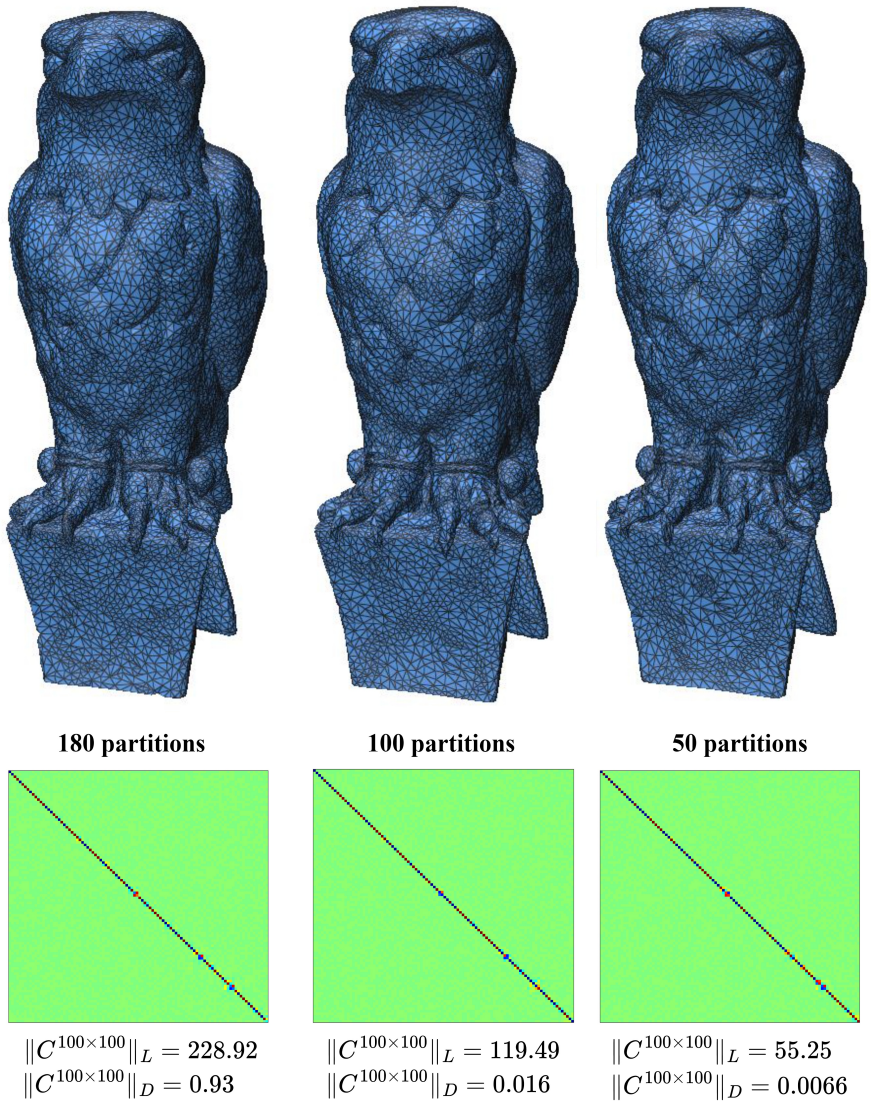


Figure 4.4: Results obtained by employing different number of partitions on eagle mesh

Figure 4.4 presents the output meshes, functional maps and norms obtained by dividing the eagle mesh into 180, 100 and 50 partitions from left to right. Here, eagle mesh is again simplified from 25727 vertices to 14000 vertices. As the number of partitions is decreased, both of the norms tend to get closer to zero, meaning that the spectrum of the Laplacian operator is better preserved. This is because while the partitions grow, their ability to represent the global properties of the overall mesh gets improved as well. Even though it is not very distinctive from the visualization of the functional maps here, if the high-frequency parts of the functional maps are examined carefully, it can be observed that the functional map resembles the diagonal matrix more and more, as it goes from left to right.

4.2.2 Number of Eigenvalues

The timing results obtained by preserving different number of eigenvalues are provided in Figure 4.5. Here, eagle mesh is simplified from 25727 vertices to 10000 vertices. It can be seen that as the number of preserved eigenvalues is increased, the simplification time also increases, since the eigen-decomposition process performed for the Laplacian operator at each step of the algorithm takes longer time than before.

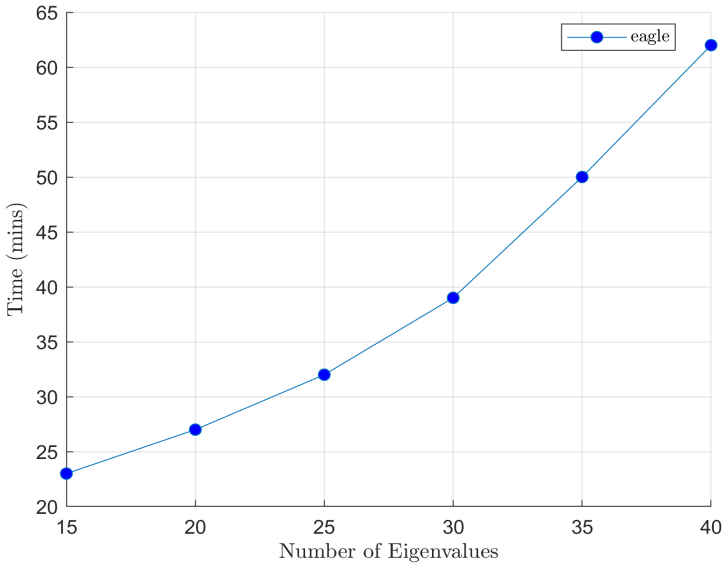


Figure 4.5: Simplification times for eagle mesh by preserving different number of eigenvalues

As stated in Section 3.1, the number of eigenvalues to preserve is selected as 15 for this method, but different values can also be specified through the input parameter k. The spectrum preservation results obtained by changing the number of eigenvalues to preserve are presented in Figure 4.6, where the bunny mesh is coarsened until 1000 vertices remain. The provided plots show that both the Laplacian commutativity and the orthonormality norms tend to decrease, as the number of eigenvalues to preserve is increased. This is because increasing the number of eigenvalues results with a better spectrum preservation for the higher frequency components. However, as previously stated by the works conducted in [28] and [26], this holds only up to a certain point. After that point, it only introduces higher computational cost to the method.

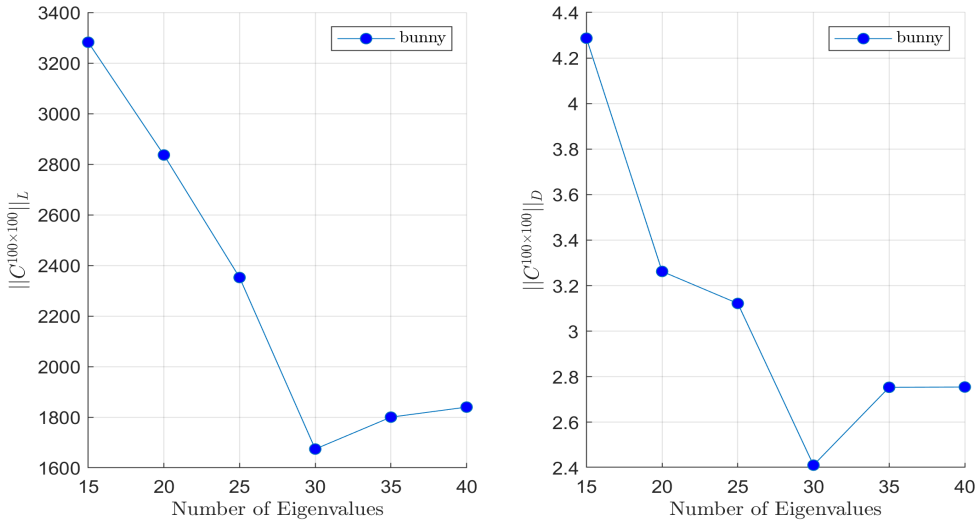


Figure 4.6: $\|\cdot\|_L$ and $\|\cdot\|_D$ values obtained on bunny mesh for different number of eigenvalues

4.2.3 Similar Edge Collapses

For the parameters x and n, different combinations are tested such that $3x < n$ (see Section 3.1.3). As a reminder, x stands for the number of similar edges to collapse, while n determines the size of the edge set that these similar collapses are chosen among. Experimentally, it is found that the best performance is mostly achieved by selecting x as 4, and n as 20.

Figure 4.7 presents the results obtained on teddy mesh (simplified from 9480 vertices

to 1000 vertices), for some of the tested combinations. In general, it is observed that as the number of similar edge collapses increases, both of the norms increase. It can also be seen that the bottom right parts of the functional maps which correspond to the high-frequency eigenvalues start to scatter, indicating that the capability to preserve the spectrum at the higher frequencies decrease. This is because as the number of similar edge collapses increases, more edges are chosen among the same restricted subset of edges. However, the mesh gets modified with each edge collapse, while the edges in the restricted subset were selected as the best choices for an older state of the mesh. At some point, the edges outside that subset may be preserving the spectrum more than the edges in the subset, however they are left as not considered. Therefore, it is a better practice not to keep the number of similar edge collapses too high.

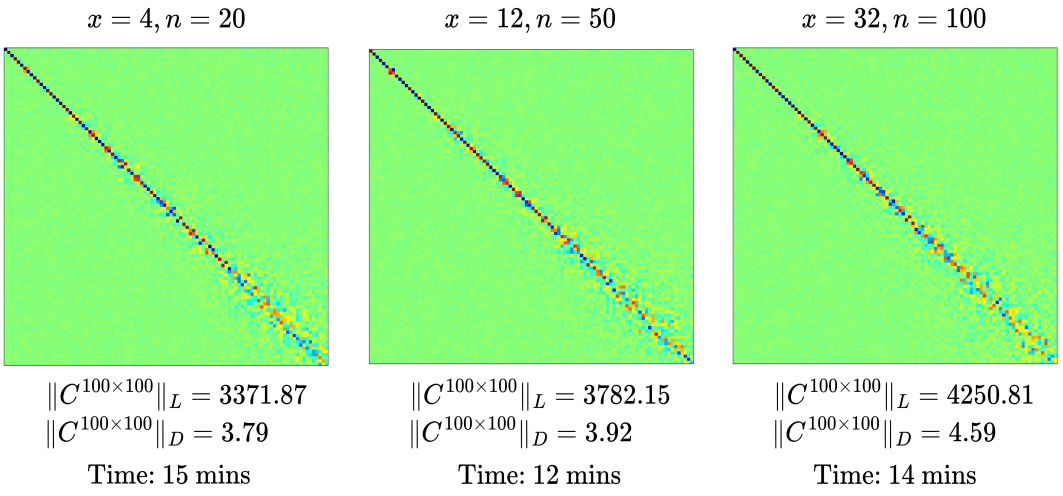


Figure 4.7: Spectrum preservation and timing results for various values of x and n

The timing results can also be found at the bottom part of the Figure 4.7, for the related x and n combinations. It can be observed that increasing the number of similar edge collapses decrease the simplification time, but only up to a point. At first glance, it might be appealing to eliminate a large number of similar edges by performing less eigenvalue computations. However, it is a bit impractical to keep the size of the edge set too large, because when there are many edges to consider, timing enhancements do not meet the expectations. Instead, choosing the similar edges from a small subset produces both better quality outputs and better timings. By taking all these consider-

ations into account, for the rest of the experiments x and n are selected as 4 and 20 respectively, unless otherwise specified.

In order to emphasize the benefits of the similar edge collapse approach, the results obtained by disabling and enabling similar edge collapses are shown in Figure 4.8. It can be seen that enabling similar edge collapses reduces the simplification time to approximately one fifth of the previous timing, while affecting the spectrum as little as possible.

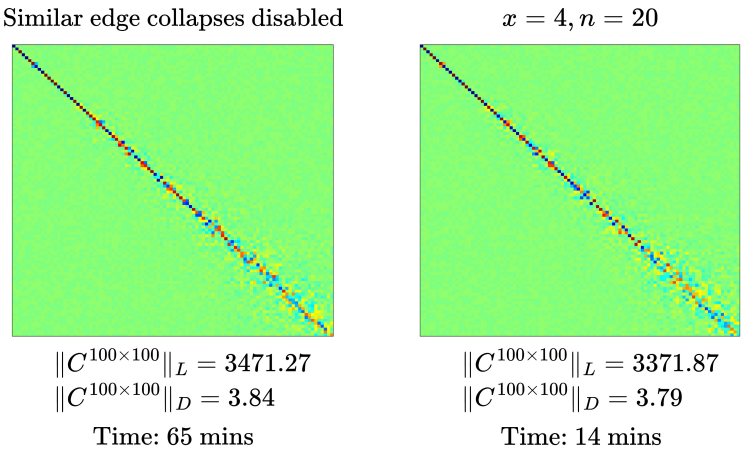


Figure 4.8: Results obtained by enabled and disabling similar edge collapses

4.2.4 Threading

For several meshes, timing results obtained by enabling and disabling threading are provided in Table 4.1.

Table 4.1: Timings obtained by enabling and disabling threading

Mesh	Number of Vertices	Number of Threads	Time with Threading (mins)	Time without Threading (mins)
bunny	3485	15	9	18
camel	9757	48	13	40
cactus	25131	180	17	75
eagle	25727	180	18	76

Here, the number of threads equals the number of partitions that the mesh is divided into. As can be seen, threading provides a significant reduction in simplification times. However, the acceleration amount is not the same as the number of threads. This is because threads are executed batch by batch, in a way that only the partitions that are not direct neighbors of each other are run concurrently.

Although there is no linear relationship between the total number of partitions and the amount of acceleration, contributions of threading become more noticeable for the meshes with more partitions. The reason behind this can be visualized with a plot as in Figure 4.9. Green, blue and red lines respectively indicate the minimum, average and maximum number of threads existing in a batch, for the given number of partitions. It can be observed that when there are more partitions in a mesh, the average number of threads per batch gets increased as well, leading to a higher amount of parallelization. For instance, if there are 8 partitions in a mesh, the number of threads in each batch will be between 2 and 3. On the other hand, if there are 100 partitions in a mesh, the number of threads per batch will range from 11 to 24, which results with a more remarkable timing reduction.

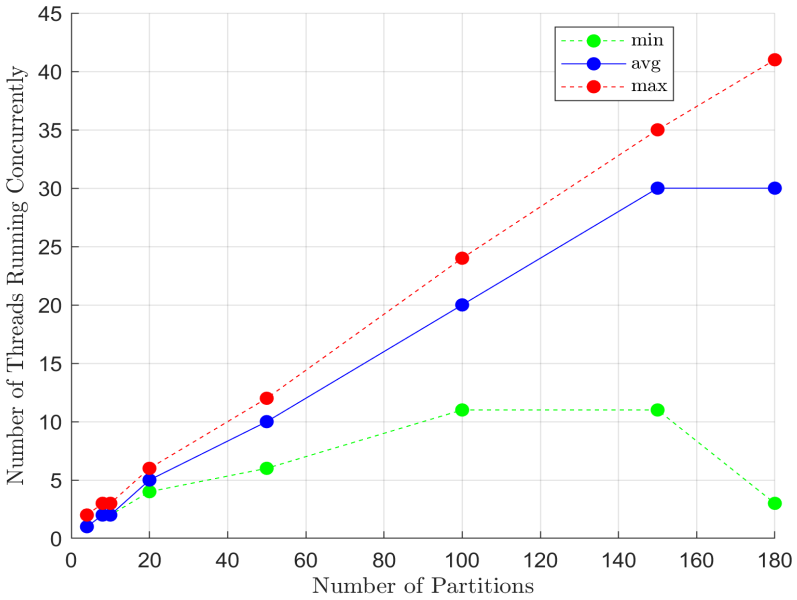


Figure 4.9: The number of threads (partitions) running at the same time for various number of partitions

4.2.5 Comparisons

The proposed method is evaluated by comparing against some of the existing spectrum-preserving methods such as spectral coarsening [24], spectral simplification [26] and chordal coarsening [27]. For fair comparison, all of the methods are configured to preserve the first 15 components of the spectrum along with ours. It should be noted that our method and the spectral mesh simplification method proposed by Lescoat et al. [26] produce a coarsened mesh as output. However, the methods proposed by Liu et al. [24] and Chen et al. [27] do not produce an output mesh. They are based on coarsening the supplied differential operator instead of a mesh. Therefore, for their methods, the vertices that are selected to exist on the coarsened domain are visualized by marking the vertices on the original mesh.

To make the quantitative results comparable, before obtaining the results, all the meshes are scaled so that their surface areas correspond to unit area. For the mass matrices M and $\widetilde{M}$, it is ensured that $tr(M) = tr(\widetilde{M}) = 1$. This surface area normalization is also required to be able to compare the eigenvalues robustly, since they are affected by the surface area [34].

For all the results, the functional maps are calculated for the eigenvectors corresponding to the first 100 eigenvalues and the related norms are presented. In addition, for each mesh, the first 100 eigenvalues of the coarsened Laplacian operator obtained with different methods are compared with a plot, indicating how much they resemble the original eigenvalues.

By examining the norms provided for the functional map $C^{100\times100}$, it can be observed that the presented method outperforms the others when considering the first 100 eigenvalues. Even though the algorithm is focused on preserving the first 15 eigenvalues, it is shown that it manages to maintain the spectrum beyond the first 15. In some cases as in Figure 4.10, our results are head-to-head with Lescoat et al. [26], but for most of the cases, it can be seen that this method preserves the spectrum better than theirs. When the appearance of the output meshes are compared, their method produces meshes with better quality triangulations, with respect to the presented partition based method. However, Figure 4.20 shows that, in some cases such as the head

and the toes of the camel mesh, the details may get completely lost with their method, while this method still manages to preserve them.

When the presented method is compared with the method of Liu et al. [24], from both the functional maps and the eigenvalue plots, it can be observed that the presented method preserves the spectrum better than theirs, while considering a small number of eigenvalues.

When the functional map visualizations are considered, for the majority of the cases, the method proposed by Chen et al. [27] seems to preserve the spectrum better than all other methods including ours, since they resemble the identity matrix more. However, it should be noted that the functional map visualizations represent the eigenvector preservation. When the norms are examined, it can be seen that their Laplacian commutativity norms are much larger than ours. This is because although their method manages to preserve the first 15 eigenvalues, it fails to preserve the eigenvalues beyond that point, as can be seen from the eigenvalue plots.

The main advantage of this method over the existing methods is to be able to preserve the spectrum by focusing on a small number of eigenvalues such as 15, while outputting a simplified mesh. This method focuses on preserving the eigenvalues directly, whereas other existing methods carry out this implicitly by preserving the eigenvectors. In theory, working with eigenvalues might seem to be more efficient than working with eigenvectors. Unfortunately, since this method contains many eigenvalue computations from scratch, the timings are much longer than other methods.

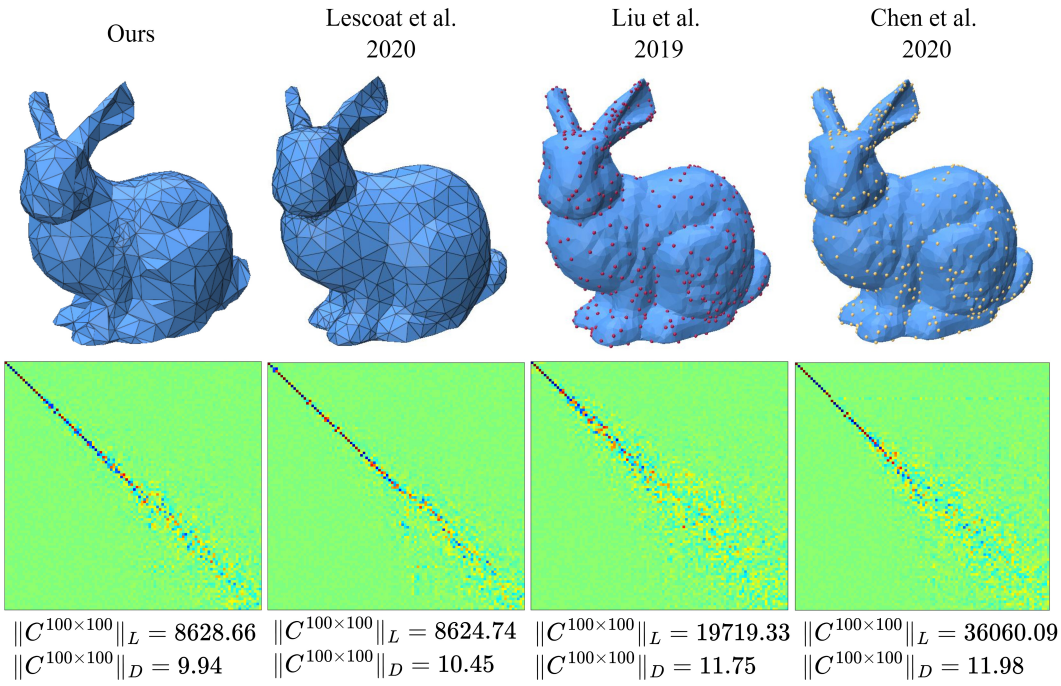


Figure 4.10: With each method, bunny mesh is simplified from 3485 to 600 vertices (17% of its initial size).

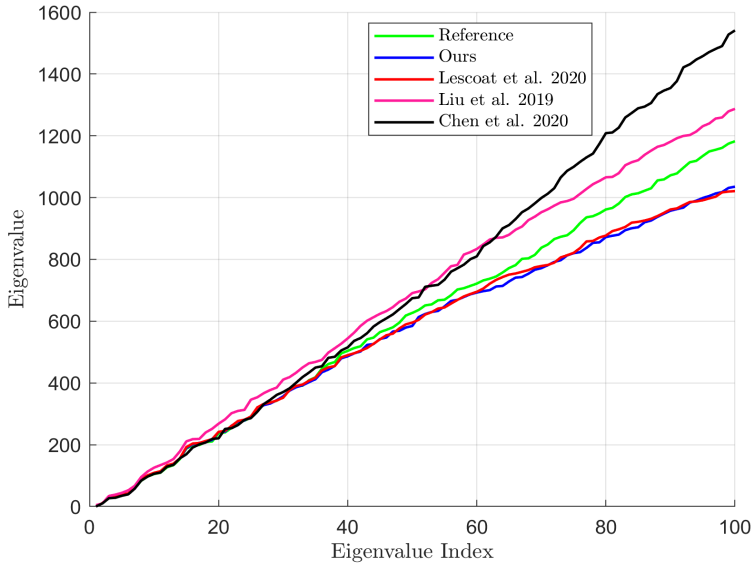


Figure 4.11: The first 100 eigenvalues of the Laplacian operator for the coarsened bunny mesh

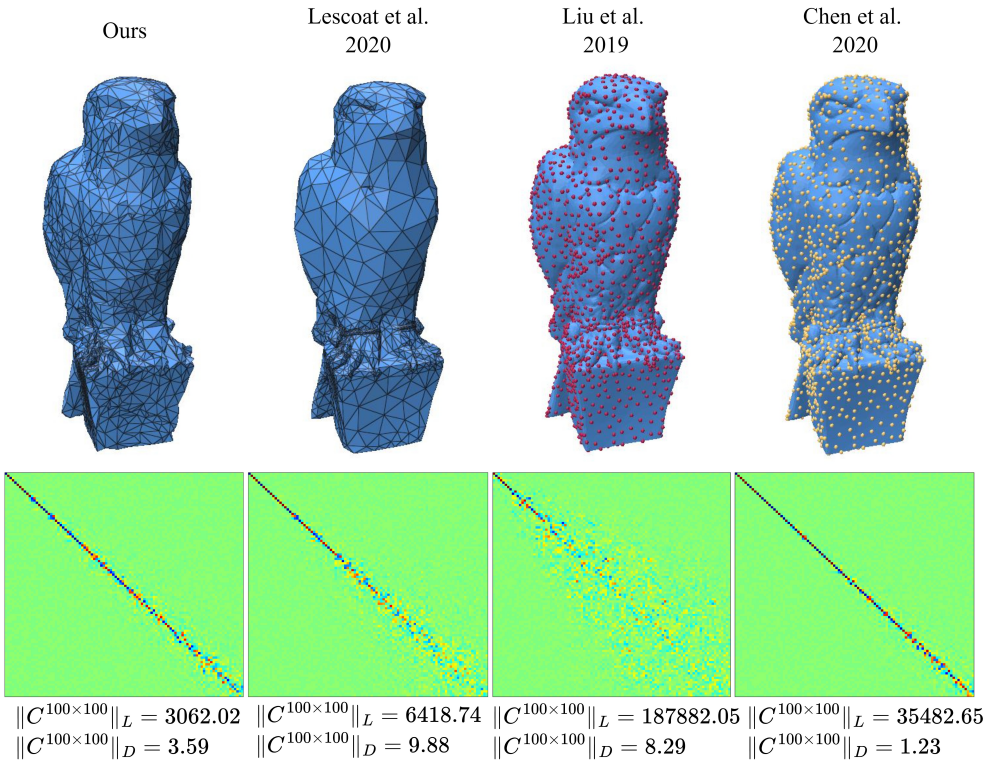


Figure 4.12: With each method, eagle mesh is simplified from 25727 to 2000 vertices (8% of its initial size).

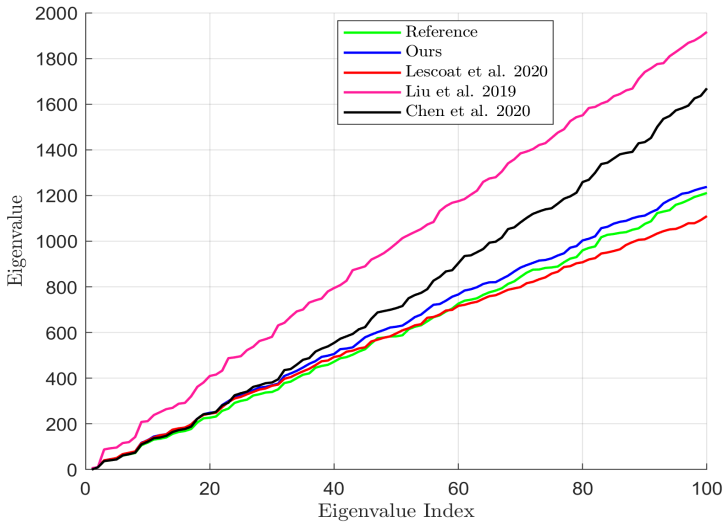


Figure 4.13: The first 100 eigenvalues of the Laplacian operator for the coarsened eagle mesh

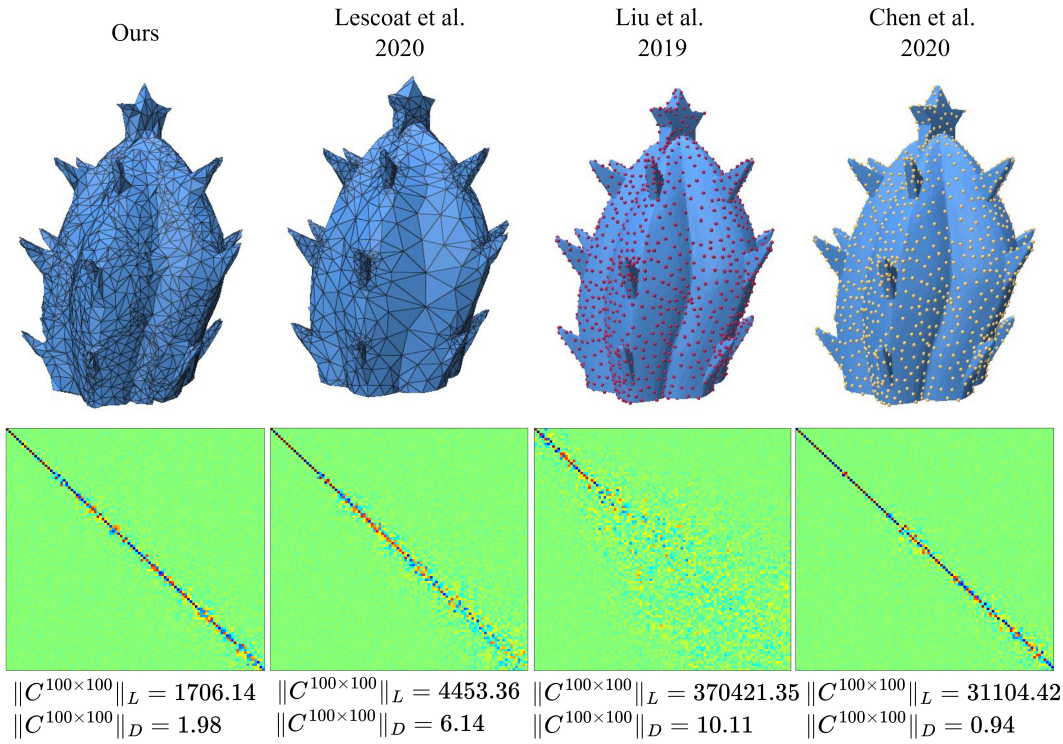


Figure 4.14: With each method, cactus mesh is simplified from 25131 to 2000 vertices (8% of its initial size).

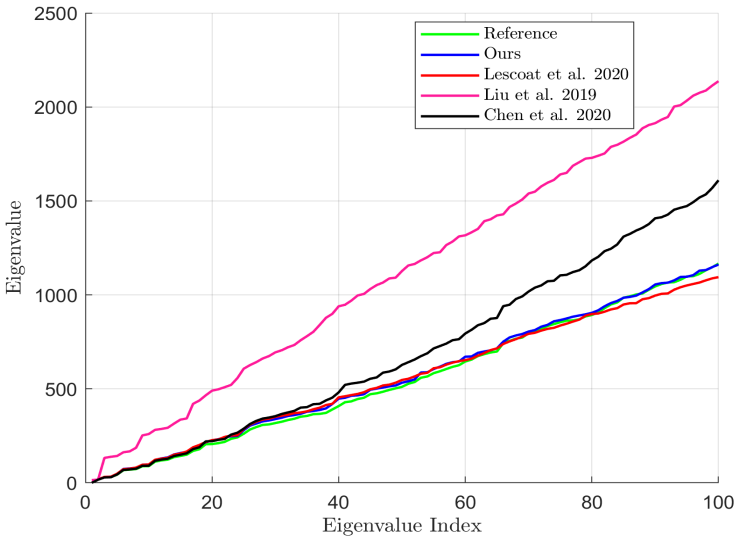


Figure 4.15: The first 100 eigenvalues of the Laplacian operator for the coarsened cactus mesh

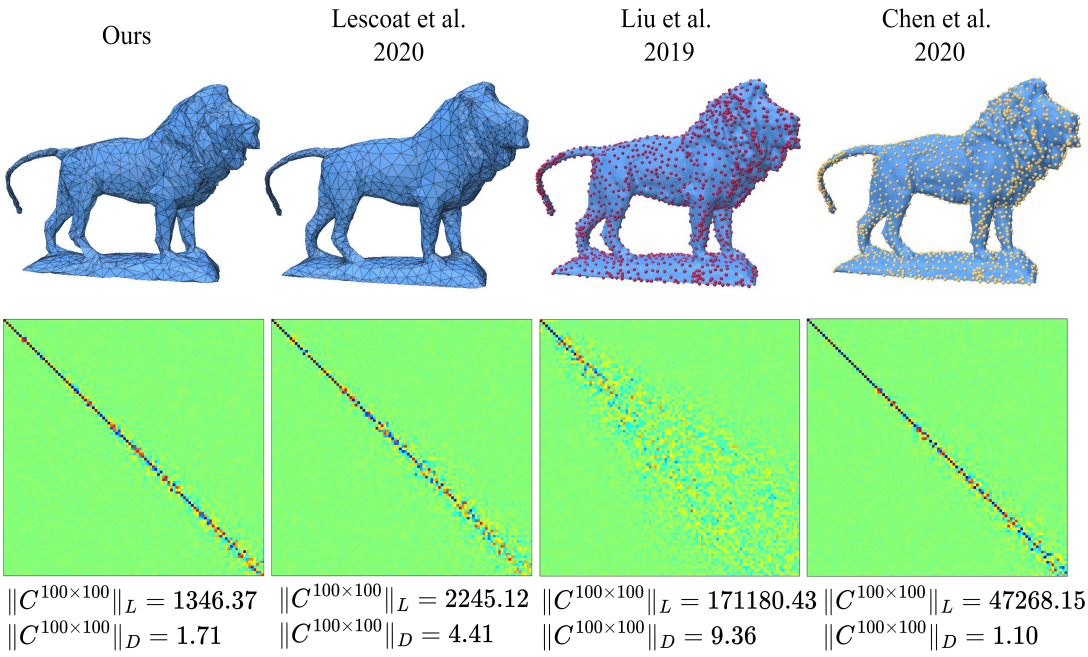


Figure 4.16: With each method, lion mesh is simplified from 20212 to 2000 vertices (10% of its initial size).

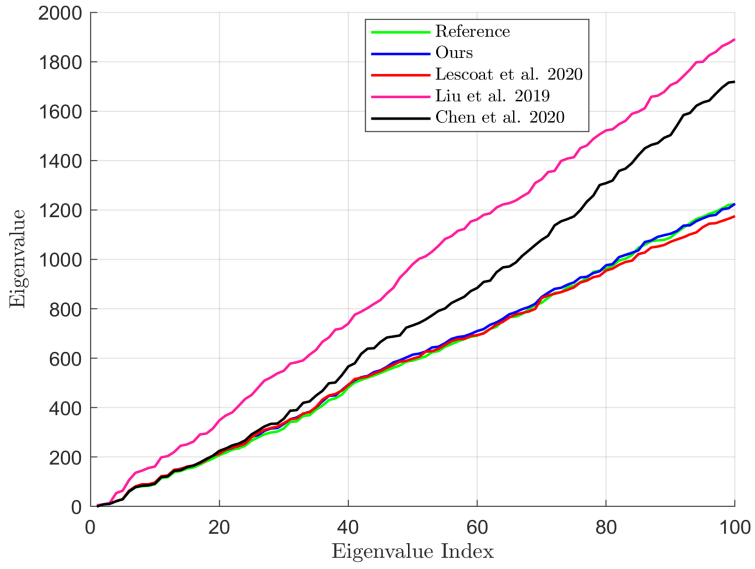


Figure 4.17: The first 100 eigenvalues of the Laplacian operator for the coarsened lion mesh

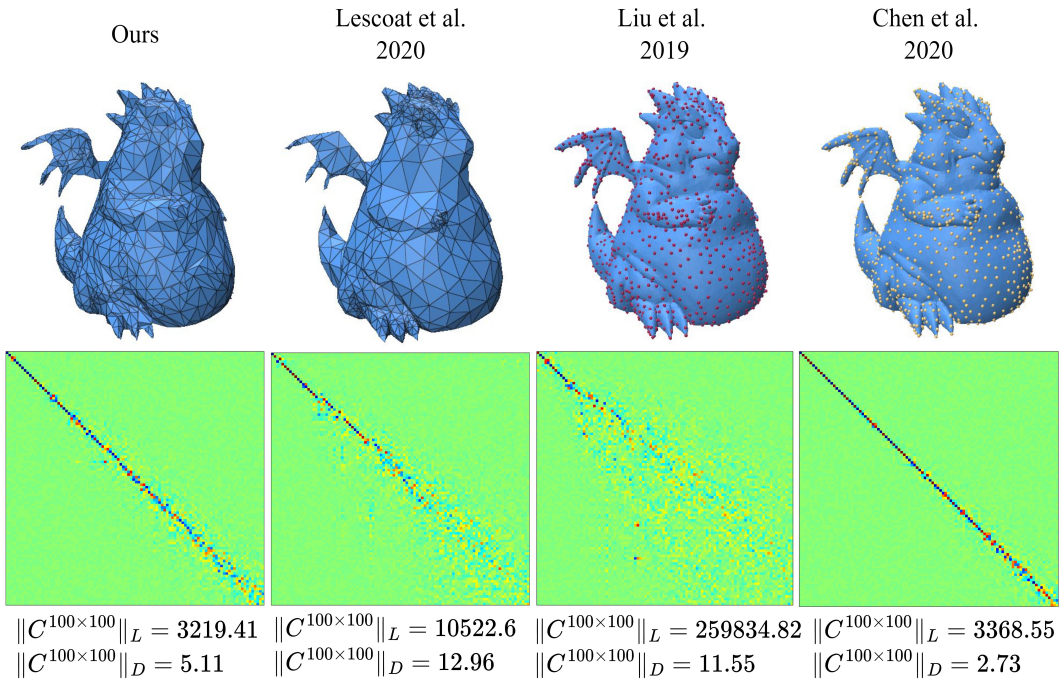


Figure 4.18: With each method, dragon mesh is simplified from 20741 to 1500 vertices (7% of its initial size).

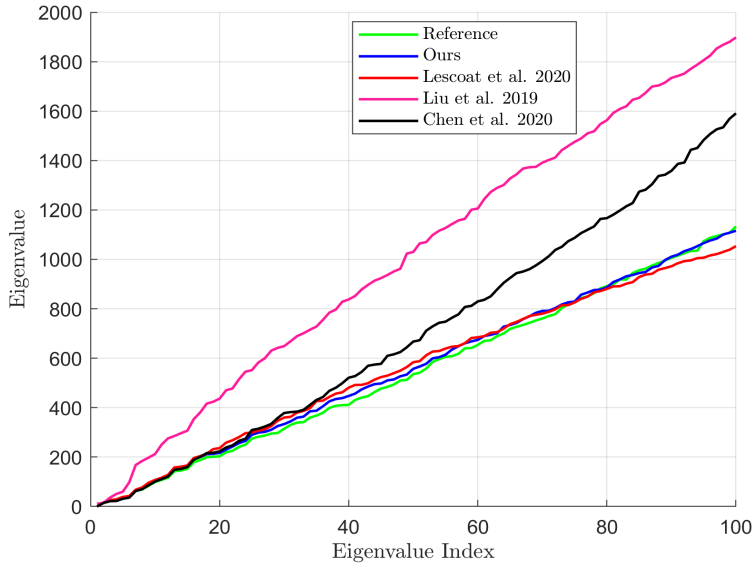


Figure 4.19: The first 100 eigenvalues of the Laplacian operator for the coarsened dragon mesh

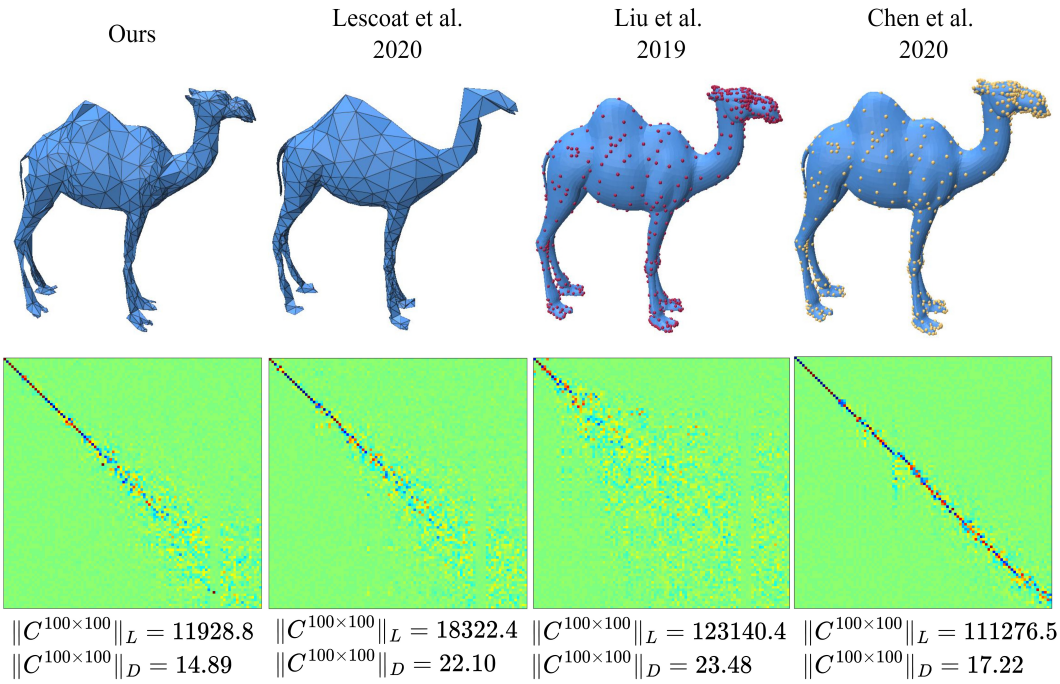


Figure 4.20: With each method, camel mesh is simplified from 9757 to 800 vertices (8% of its initial size).

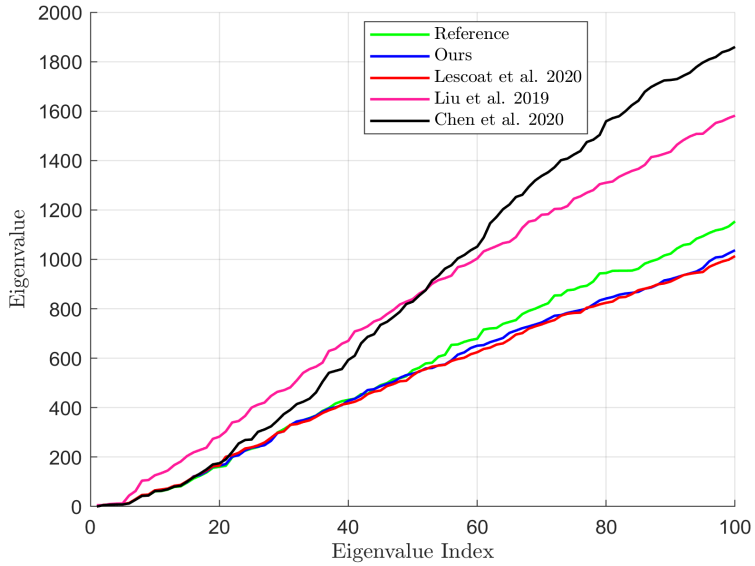


Figure 4.21: The first 100 eigenvalues of the Laplacian operator for the coarsened camel mesh

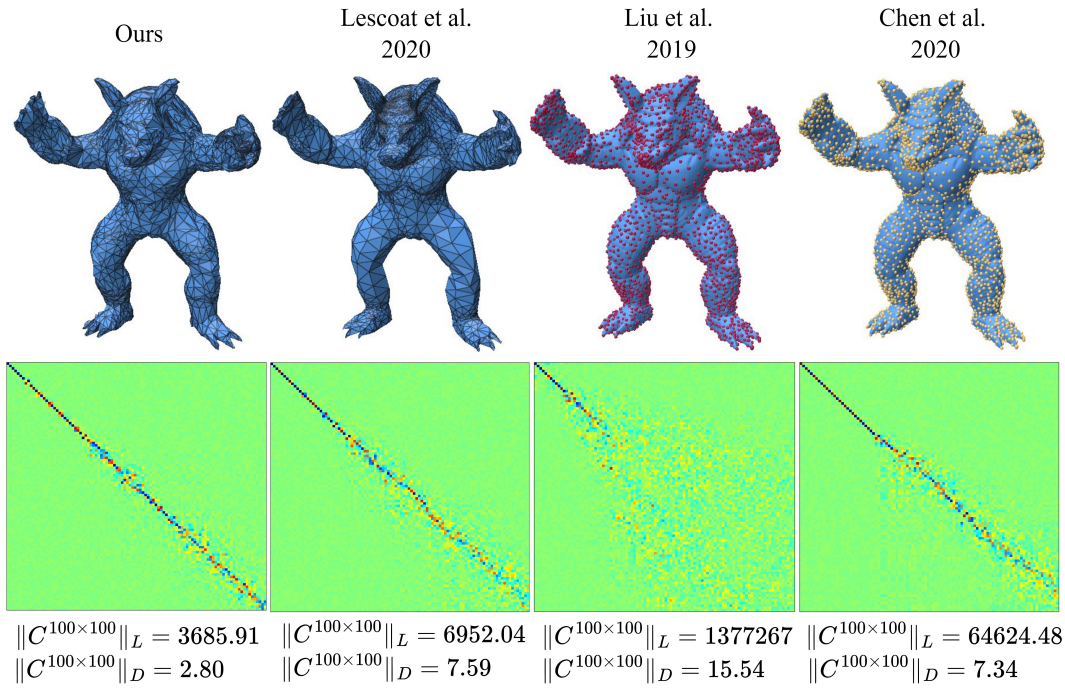


Figure 4.22: With each method, armadillo mesh is simplified from 49990 to 3000 vertices (6% of its initial size).

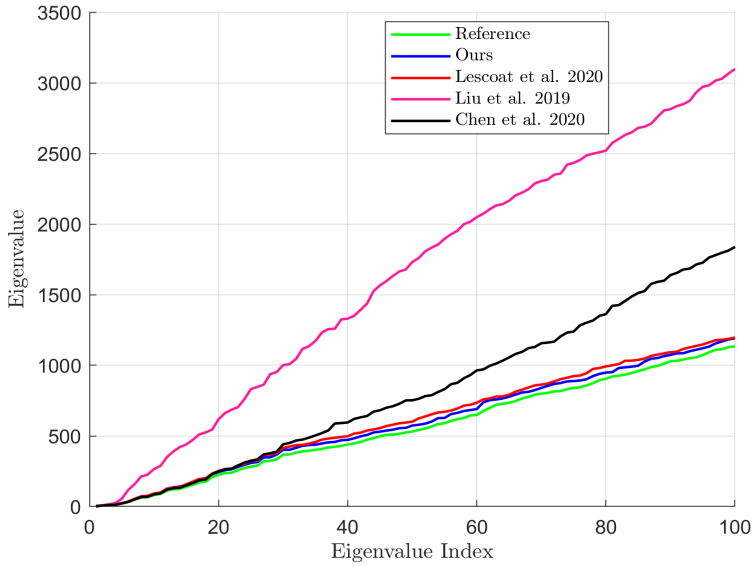


Figure 4.23: The first 100 eigenvalues of the Laplacian operator for the coarsened armadillo mesh

4.2.6 Heat Kernel Signature

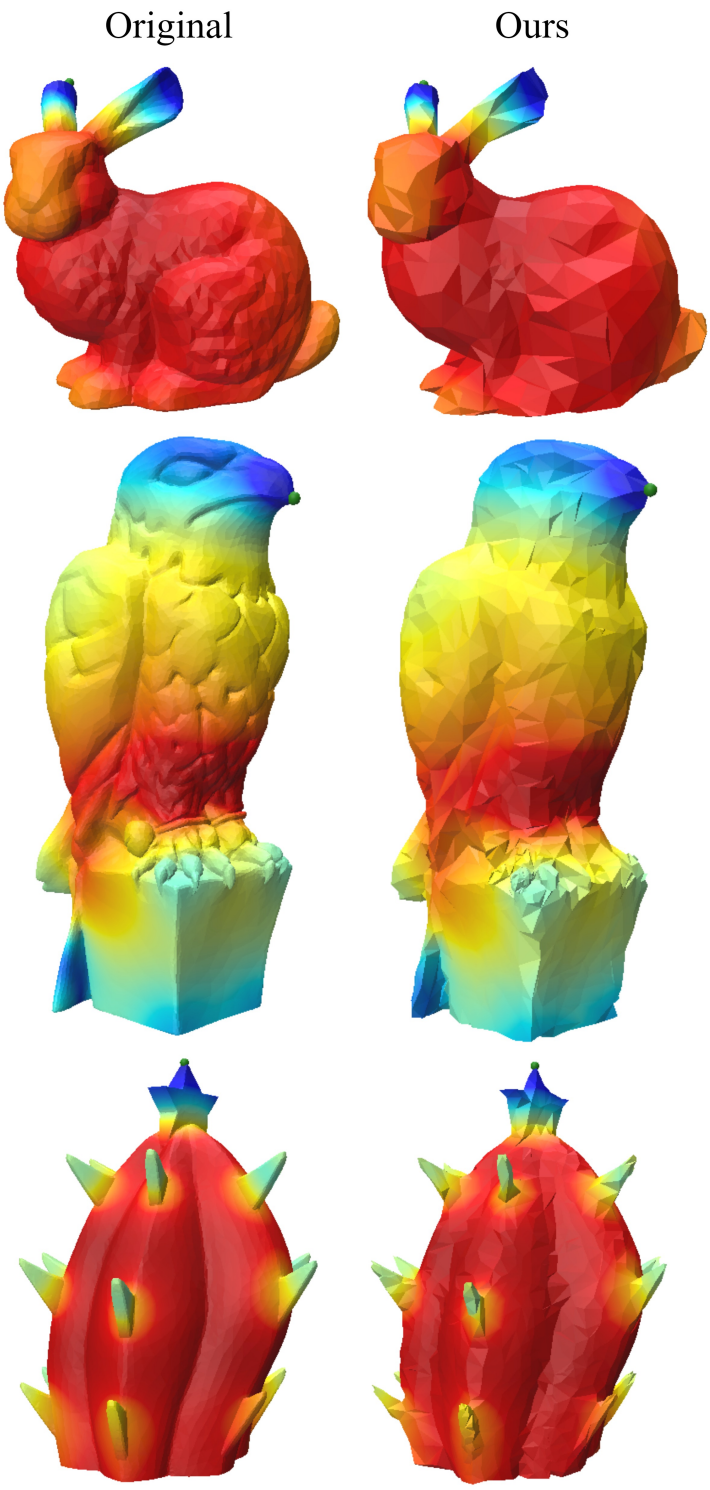


Figure 4.24: Heat kernel signature for the original meshes (left) and their simplified versions (right)

For an application example, the visualizations of the heat kernel signature on the original high resolution mesh and the coarsened version are provided in Figure 4.24, for several meshes. For each vertex, the heat kernel signatures are visualized with respect to a reference vertex, which is marked with green in the figure. Here, the bunny mesh is simplified to 17% of its initial size, while the others are simplified to 8%. It can be observed that even with a high reduction ratio, the heat kernel signatures almost stay the same, which implies the spectrum preservation between the original and the simplified meshes.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this thesis, a mesh simplification method, which is focused on preserving the spectral properties of the input mesh as much as possible, is presented. The main idea lying at the core of the algorithm is comparing the eigenvalues of the Laplacian operators constructed on the mesh at different steps, to decide which edges to collapse. In order to reduce the computational cost caused by the eigenvalue decompositions carried out in the algorithm, the mesh is divided into smaller partitions and each partition is coarsened separately. On top of that, to minimize the number of eigenvalue decompositions performed and to further improve the simplification times, threading and an approach based on eliminating spectrally close edges are introduced. The algorithm is tested on several meshes. By utilizing the functional maps and quantitative metrics proposed by Liu et al. [24] and Lescoat et al. [26], it is demonstrated that the proposed method is capable of maintaining the spectrum by preserving a small number of eigenvalues. The method is compared with the existing spectral coarsening methods and it is shown that the presented method is able to preserve the spectrum at least as successfully as the others. It is demonstrated that the coarsened meshes produced by the algorithm are still able to represent the heat kernel signature similar to the way it was represented on the original mesh. Moreover, it is also shown that spectrum-preserving mesh simplification can be performed only by considering the eigenvalues, without needing the eigenvectors. Therefore, it is believed that this method will benefit the future works, whose main focus is spectrum preservation.

The main limitation of this algorithm is about efficiency. It requires the computation of the eigenvalues from scratch for each edge collapse that is considered. Even with the methods introduced with the aim of reducing the number of eigenvalue computa-

tions, the algorithm is still unable to reach the simplification times of other existing methods. One way to overcome the efficiency problem would be adapting the classic priority queue method introduced by Garland and Heckbert [1] with the presented cost metric, just as Lescoat et al. [26] did. However, to do this, an update rule for the edge collapse costs should be found, such that after an edge is collapsed, the costs of the remaining edges are updated without needing to compute the eigenvalues again. In this way, the burden caused by the eigenvalue computations would be significantly reduced.

Another limitation of this method is that for an edge collapse, the algorithm selects the merged vertex position as the midpoint of the edge. Once the timings are improved, the merged vertex position which results with the best spectrum preservation can be computed by considering several different positions.

As a final note, this method is a greedy algorithm, so it does not guarantee that the set of edge collapses performed leads to the optimal preservation of the spectrum. Yet, it is believed that this naive way of utilizing the eigenvalues of the Laplacian operator may establish a ground for more advanced spectrum-preserving algorithms.

REFERENCES

[1] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 209–216, 1997.

[2] O. Sorkine, "Laplacian mesh processing," *Eurographics (State of the Art Reports)*, vol. 4, 2005.

[3] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy, *Polygon mesh processing*. CRC press, 2010.

[4] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen, "Decimation of triangle meshes," in *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pp. 65–70, 1992.

[5] J. Rossignac and P. Borrel, "Multi-resolution 3d approximations for rendering complex scenes," in *Modeling in computer graphics*, pp. 455–465, Springer, 1993.

[6] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Mesh optimization," in *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pp. 19–26, 1993.

[7] H. Hoppe, "Progressive meshes," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 99–108, 1996.

[8] G. Taubin, "A signal processing approach to fair surface design," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 351–358, 1995.

[9] U. Pinkall and K. Polthier, "Computing discrete minimal surfaces and their conjugates," *Experimental mathematics*, vol. 2, no. 1, pp. 15–36, 1993.

[10] H. Zhang, O. Van Kaick, and R. Dyer, "Spectral mesh processing," in *Computer graphics forum*, vol. 29, pp. 1865–1894, Wiley Online Library, 2010.

[11] R. M. Rustamov *et al.*, "Laplace-beltrami eigenfunctions for deformation invariant shape representation," in *Symposium on geometry processing*, vol. 257, pp. 225–233, 2007.

[12] M. Garland and P. S. Heckbert, "Simplifying surfaces with color and texture using quadric error metrics," in *Proceedings Visualization'98 (Cat. No. 98CB36276)*, pp. 263–269, IEEE, 1998.

[13] H. Hoppe, "New quadric metric for simplifying meshes with appearance attributes," in *Proceedings Visualization'99 (Cat. No. 99CB37067)*, pp. 59–510, IEEE, 1999.

[14] P. Cignoni, C. Montani, and R. Scopigno, "A comparison of mesh simplification algorithms," *Computers & Graphics*, vol. 22, no. 1, pp. 37–54, 1998.

[15] H. Zhang, R. Liu, *et al.*, "Mesh segmentation via recursive and visually salient spectral cuts," Citeseer.

[16] R. Liu and H. Zhang, "Mesh segmentation via spectral embedding and contour analysis," in *Computer Graphics Forum*, vol. 26, pp. 385–394, Wiley Online Library, 2007.

[17] V. Jain, H. Zhang, and O. Van Kaick, "Non-rigid spectral correspondence of triangle meshes," *International Journal of Shape Modeling*, vol. 13, no. 01, pp. 101–124, 2007.

[18] Z. Karni and C. Gotsman, "Spectral compression of mesh geometry," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 279–286, 2000.

[19] M. Reuter, F.-E. Wolter, and N. Peinecke, "Laplace-spectra as fingerprints for shape matching," in *Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pp. 101–106, 2005.

[20] M. Reuter, F.-E. Wolter, and N. Peinecke, "Laplace–beltrami spectra as 'shape-dna'of surfaces and solids," *Computer-Aided Design*, vol. 38, no. 4, pp. 342–366, 2006.

[21] J. Sun, M. Ovsjanikov, and L. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," in *Computer graphics forum*, vol. 28, pp. 1383–1392, Wiley Online Library, 2009.

[22] M. Ovsjanikov, J. Sun, and L. Guibas, "Global intrinsic symmetries of shapes," in *Computer graphics forum*, vol. 27, pp. 1341–1348, Wiley Online Library, 2008.

[23] A. C. Öztireli, M. Alexa, and M. Gross, "Spectral sampling of manifolds," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 6, pp. 1–8, 2010.

[24] H.-T. D. Liu, A. Jacobson, and M. Ovsjanikov, "Spectral coarsening of geometric operators," *arXiv preprint arXiv:1905.05161*, 2019.

[25] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. Guibas, "Functional maps: a flexible representation of maps between shapes," *ACM Transactions on Graphics (ToG)*, vol. 31, no. 4, pp. 1–11, 2012.

[26] T. Lescoat, H.-T. D. Liu, J.-M. Thiery, A. Jacobson, T. Boubekeur, and M. Ovsjanikov, "Spectral mesh simplification," in *Computer Graphics Forum*, vol. 39, pp. 315–324, Wiley Online Library, 2020.

[27] H. Chen, H.-T. D. Liu, A. Jacobson, and D. I. Levin, "Chordal decomposition for spectral coarsening," *arXiv preprint arXiv:2009.02294*, 2020.

[28] V. Jain and H. Zhang, "A spectral approach to shape-based retrieval of articulated 3d models," *Computer-Aided Design*, vol. 39, no. 5, pp. 398–407, 2007.

[29] D. S. Johnson and M. R. Garey, *Computers and intractability: A guide to the theory of NP-completeness*. WH Freeman, 1979.

[30] G. Karypis and V. Kumar, "A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices," *University of Minnesota, Department of Computer Science and Engineering, Army HPC Research Center, Minneapolis, MN*, vol. 38, 1998.

[31] D. J. Welsh and M. B. Powell, "An upper bound for the chromatic number of a graph and its application to timetabling problems," *The Computer Journal*, vol. 10, no. 1, pp. 85–86, 1967.

[32] G. Guennebaud, B. Jacob, *et al.*, “Eigen v3.” http://eigen.tuxfamily.org, 2010.

[33] Y. Qiu, “Spectra. a library for large scale eigenvalue problems.” https://spectralib.org/, 2018.

[34] M. Reuter, F.-E. Wolter, M. Shenton, and M. Niethammer, “Laplace–beltrami eigenvalues and topological features of eigenfunctions for statistical shape analysis,” *Computer-Aided Design*, vol. 41, no. 10, pp. 739–755, 2009.