EUROGRAPHICS 2020 / U. Assarsson and D. Panozzo
(Guest Editors)

Volume 39 (2020), Number 2

Spectral Mesh Simplification

Thibault Lescoat[1] Hsueh-Ti Derek Liu[2] Jean-Marc Thiery[1] Alec Jacobson[2] Tamy Boubekeur[4,1] Maks Ovsjanikov[3]

[1] LTCI, Télécom Paris, Institut Polytechnique de Paris, France [2] University of Toronto, Canada [3] École Polytechnique, France [4] Adobe

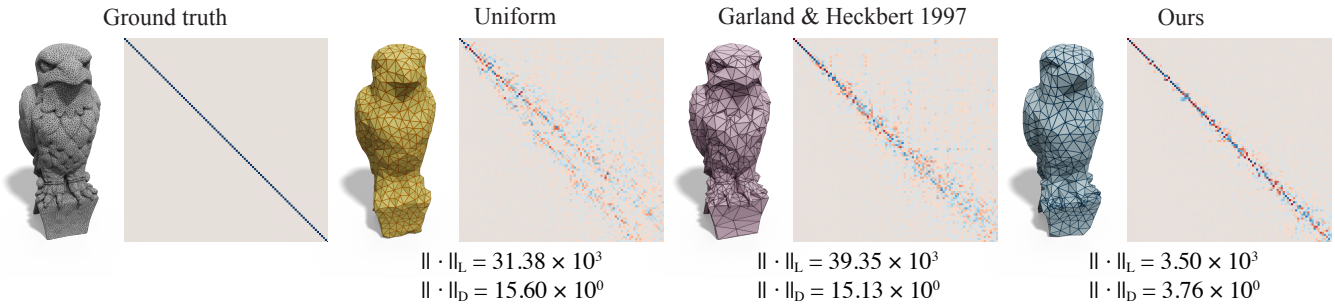


Figure 1: *We propose to simplify a mesh using edge collapses while aiming to preserve the input eigenvectors and eigenvalues as much as possible. While different strategies exist to reduce a mesh (here, from 25,727 vertices to 771 vertices, or 3% of its initial size), such as enforcing uniform edge lengths or using the Quadric Error Metric [GH97], they do not focus on keeping the spectral properties of the mesh. Reducing a mesh can be spectrally described using functional maps [OBCS*12], shown here with the output meshes, and which should ideally be diagonal. We also evaluate functional maps using two norms, the Laplacian commutativity $\|\cdot\|_L$ and the orthogonality $\|\cdot\|_D$.*

Abstract

The spectrum of the Laplace-Beltrami operator is instrumental for a number of geometric modeling applications, from processing to analysis. Recently, multiple methods were developed to retrieve an approximation of a shape that preserves its eigenvectors as much as possible, but these techniques output a subset of input points with no connectivity, which limits their potential applications. Furthermore, the obtained Laplacian results from an optimization procedure, implying its storage alongside the selected points. Focusing on keeping a mesh instead of an operator would allow to retrieve the latter using the standard cotangent formulation, enabling easier processing afterwards. Instead, we propose to simplify the input mesh using a spectrum-preserving mesh decimation scheme, so that the Laplacian computed on the simplified mesh is spectrally close to the one of the input mesh. We illustrate the benefit of our approach for quickly approximating spectral distances and functional maps on low resolution proxies of potentially high resolution input meshes.

1. Introduction

Triangle meshes remain a predominant representation of 3D surfaces. When the complexity of a given mesh exceeds computational resources, we rely on mesh simplification methods to remove vertices, edges, and faces. In rendering, efficient simplification methods can dramatically reduce the complexity of a mesh without affecting its appearance. It is tempting to repurpose appearance-preserving simplification methods for other geometry processing tasks.

Unfortunately, appearance-based methods do not preserve the spectral properties of the important differential operators upon which much of modern geometry processing is built (see Figure 1). As a result, solutions computed on such a coarse mesh can be incorrect or misleading. Alternatively, previous coarsening methods that do preserve spectral properties work purely algebraically on the operator matrices and do not produce a geometric mesh (see Figure 2). The lack of a mesh limits the use of coarsening in many downstream geometry processing tasks.

We present the first mesh simplification method intentionally designed to preserve spectral properties. We propose adapting the standard greedy edge-collapse mesh-simplification algorithm with a novel cost function that measures spectral preservation of a given operator (e.g., the cotangent Laplacian). Unlike algebraic methods that directly output a reduced operator (i.e., matrix), our method outputs a manifold triangle mesh with 3D vertex positions. Reconstructing the operator on the output mesh will preserve both the eigenvalues and eigenvectors of the operator on the input mesh.

Confirmed by a series of experiments, our method preserves spectral properties nearly as well as purely algebraic methods, while still outputting an embedded mesh like standard simplification algorithms. We demonstrate our approach's effectiveness for geodesic distance approximation and functional maps correspondence.

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

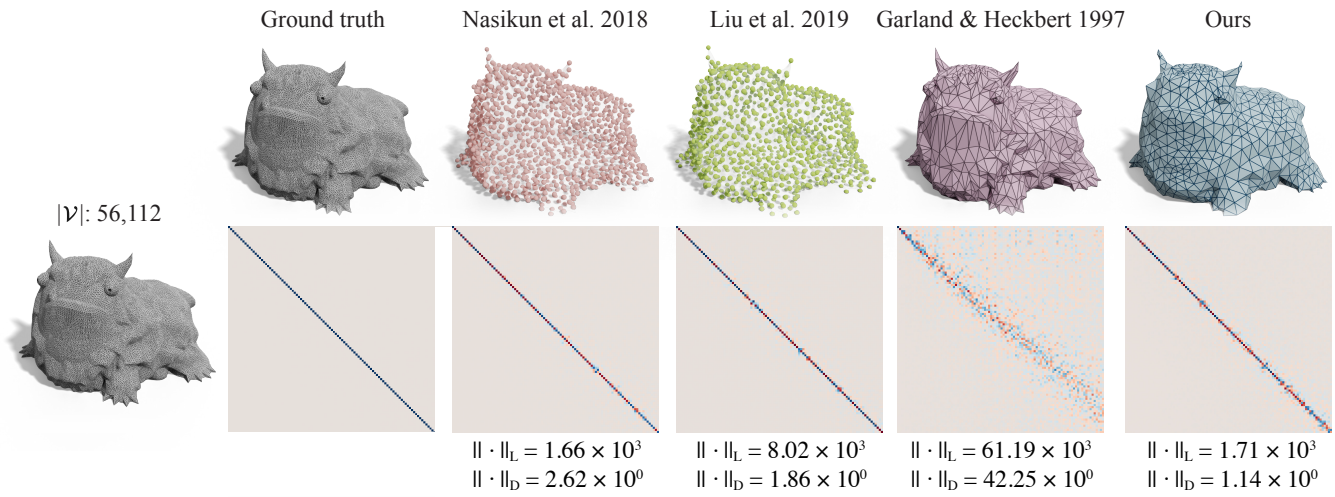


Figure 2: *Reduction from 56,112 to 1,122 vertices (2% of input size). Nasikun et al. [NBH18] approximate the original Laplacian on a subset of vertices obtained via Poisson-disk sampling. Liu et al. [LJO19] optimize the sampling and the operator, that they define on samples' 3-rings. Instead, by outputting a mesh, further processing can use a standard cotangent weighting scheme without knowledge of the reduction step.*

2. Background

Our method builds on the long history of research in mesh simplification and recent developments in spectral coarsening for differential operators. We focus the attention of this related work section on methods directly related in methodology or intention.

Classic methods for mesh simplification are based on preserving the rendered appearance of the geometric surface [SZL*92, PH97, GH97, HDD*93, CSAD04]. These methods were extended to account for other signals stored on the mesh beyond geometry including texture coordinates and colors [CMO97, GH98, Hop99, LFJG17]. Similar to many of these methods, we adapt the per-edge cost function of the basic greedy edge-collapse approach introduced by Garland & Heckbert [GH97]. Instead of optimizing perceptual metrics, we optimize a spectral metric. Spectral preservation relies on maintaining *intrinsic* properties of the surface (the metric). Previous methods have focused on maintaining *extrinsic* properties, such as keeping the coarse mesh within a small envelope around the input [CVM*96, ZG02] or strictly containing the input [SGG*00, SVJ15]. In a rare previous example of spectral mesh simplification, Li et al. [LFZ15] append modal displacement vectors for sound simulation as extra dimensions during greedy edge-collapse. However, this method preserves only the specific modes chosen and would not scale beyond a small number of frequencies. Our efficient method preserves a large span of low frequency eigenmodes and corresponding values.

Mesh simplification is closely related to graph reduction. In this more general and less constrained context, recent works have investigated spectral preservation with theoretical guarantees [KS16, Lou19, LV18]. Liu et al. [LJO19] recently demonstrated superiority over [KS16] when applied to mesh Laplacians from geometry processing. Similarly, coreset selection algorithms aim to preserve statistics or properties of a larger point set or distribution [HCB16, CS18].

An alternative approach to preserving properties of the operator built from the input mesh is to directly simplify it as a matrix. Recently, Liu et al. [LJO19] present a state-of-the-art method for spectral preservation during algebraic coarsening of common operators used in geometry processing. We refer the reader to this recent work for a comprehensive review of previous algebraic and numerical coarsening methods. Notably, Nasikun et al. [NBH18] aim at efficiently approximating eigenpairs. Both methods select a subset of points from the input; along with other recent developments (e.g., [CBO*19, OS19]), they share a common limitation compared to our method: they do not produce a mesh.

We build upon the functional maps [OBCS*12] machinery used by Liu et al. to evaluate how well a coarsening preserves spectral properties. Importantly, unlike their computationally expensive algebraic optimization, our efficient edge-collapse algorithm maintains a manifold triangle mesh.

3. Method

Our spectrum-preserving simplification method is made of two main building blocks: a simplification algorithm based on edge-collapses and a simplification metric which drives the algorithm. The metric associates a cost to any given edge-collapse, ordering them dynamically during the simplification.

3.1. Input / output

Our method takes as input a manifold triangle mesh $\mathcal{M} = (\mathcal{V}, \mathcal{F})$, which can optionally contain boundaries, and produces a simplified mesh $\widetilde{\mathcal{M}} = (\widetilde{\mathcal{V}}, \widetilde{\mathcal{F}})$, with a spectrum as close as possible to $\mathcal{M}$ when evaluating it using the standard Laplacian operator. Optionally, a coarse-to-fine restriction matrix can be produced and used when computing e.g., functional maps. We also take a unique parameter in the form of the number of eigenvectors to preserve.

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

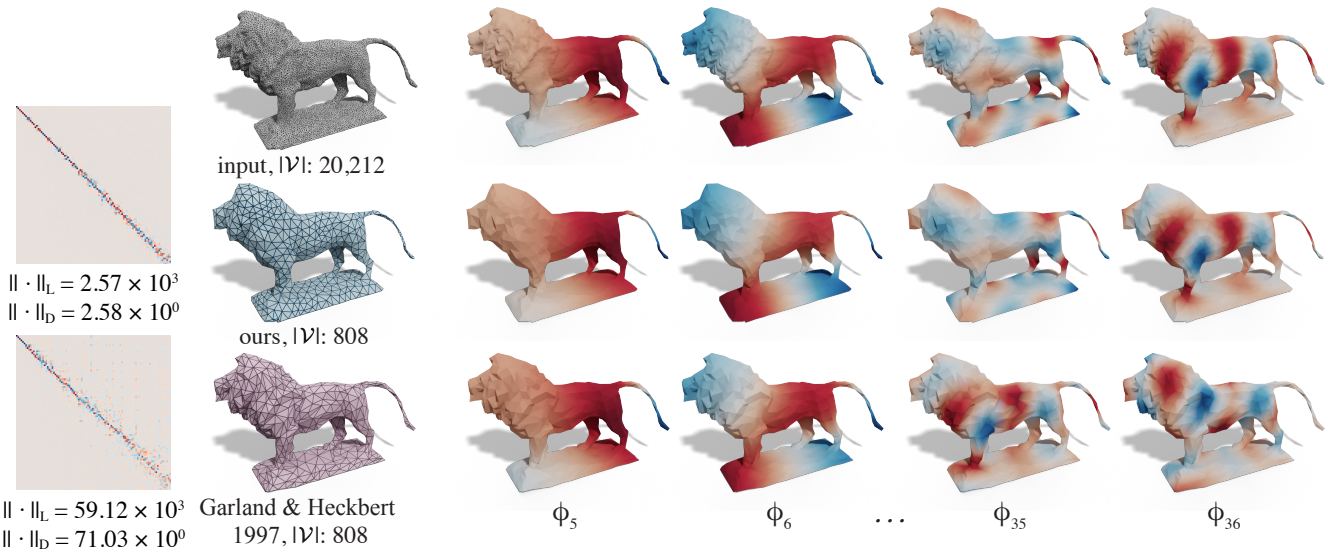


Figure 3: *While the very first eigenvectors from both our method (middle row) and QSlim [GH97] (bottom row) are very close to those of the input (top row), following eigenvectors are more sensitive; badly preserved eigenvectors exhibit patterns dissimilar to their input counterpart.*

3.2. Simplification algorithm

Given a cost for each edge of $\mathcal{M}$, our simplification algorithm (see Algorithm 1) follows the seminal idea of Garland and Heckbert [GH97]: each input edge is pushed to a priority-queue, where the priority is dictated by our specific cost metric. At all time, the edge located at the head of the queue is the next best edge to collapse i.e., with minimum cost. Once popped from the queue, the edge is collapsed, effectively removing one vertex from the mesh and resulting in a merged vertex positioned to optimize the metric. Last, the cost of the incident edges are updated, reordering the queue to respect the priority measure. This atomic mesh reduction step is iterated until reaching the desired output resolution for $\widetilde{\mathcal{M}}$, removing one vertex at a time.

Algorithm 1: Edge-collapse progressive simplification

Input: mesh $\mathcal{M} = (\mathcal{V}, \mathcal{F})$, target size N, metric $m : \mathcal{V} \times \mathcal{V} \mapsto \mathbb{R}$
Output: simplified mesh $\widetilde{\mathcal{M}} = (\widetilde{\mathcal{V}}, \widetilde{\mathcal{F}})$

$\widetilde{\mathcal{V}} \leftarrow \mathcal{V}$; $\widetilde{\mathcal{F}} \leftarrow \mathcal{F}$; `queue` $\leftarrow \{\}$;
for *edge* $e \in \mathcal{M}$ **do**
  add $(e, m(e))$ to `queue` ;
while $|\widetilde{\mathcal{V}}| > N$ *and* `queue` *not empty* **do**
  $(e, c) \leftarrow$ pop edge e with lowest cost c from `queue` ;
  collapse e (this changes $\widetilde{\mathcal{V}}$ and $\widetilde{\mathcal{F}}$) ;
  for $n \in e$*'s neighbors* **do**
    update n in `queue` ;

Optionally, we can augment this reduction algorithm to generate a restriction matrix P to be used for the computation of functional maps for instance. More precisely, when collapsing the edge (u, v), we generate the restriction matrix Q such that $V^{after} = QV^{before}$. Note that all its coefficients are positive, and its rows sum to 1. Then, with Q_i the restriction matrix of the i-th operation, the global restriction matrix is formed as follow: $P = Q_n Q_{n-1} ... Q_2 Q_1$.

3.3. Metric

While the Quadric Error Metric, introduced in the original work of Garland & Heckbert [GH97], maintain the visual appearance of the original mesh as much as possible, we propose a new alternative metric focused on spectral preservation.

Let $L, M \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ be the Laplacian and the diagonal mass matrix of $\mathcal{M}$ respectively. Similarly, $\widetilde{L}, \widetilde{M} \in \mathbb{R}^{|\widetilde{\mathcal{V}}| \times |\widetilde{\mathcal{V}}|}$ denote the Laplacian and the diagonal mass matrix of $\widetilde{\mathcal{M}}$. We note $P \in \mathbb{R}^{|\widetilde{\mathcal{V}}| \times |\mathcal{V}|}$ the fine-to-coarse restriction matrix and $\mathcal{N}_i(v)$ the i-ring of vertex v. We also use the following weighted norm:

$$\|X\|^2_{\widetilde{M}} = \mathrm{tr}(X^\top \widetilde{M} X) \tag{1}$$

We formulate the preservation of the eigenvectors of the Laplacian by the commutativity of the Laplacian and the reduction. More generally, for a signal $f \in \mathbb{R}^{|\mathcal{V}|}$, we aim for $\widetilde{M}^{-1}\widetilde{L}Pf = PM^{-1}Lf$. Thus, given K signals to preserve (here, eigenvectors of the Laplacian), represented as a matrix $F \in \mathbb{R}^{|\mathcal{V}| \times K}$, the reduction metric is:

$$E = \|P\underbrace{M^{-1}LF}_{Z} - \widetilde{M}^{-1}\widetilde{L}PF\|^2_{\widetilde{M}} \tag{2}$$

F and Z are computed only once, at the very beginning. This metric will also preserve eigenvalues, as shown in Appendix C of [LJO19].

$$\begin{array}{ccc} & \xrightarrow{M^{-1}L} & \\ f & & \bullet \\ P\downarrow & & \downarrow P \\ \bullet & \xrightarrow[\widetilde{M}^{-1}\widetilde{L}]{} & \widetilde{f} \end{array}$$

Since $\widetilde{M}$ is a diagonal matrix, the weighted norm of Equation (1)

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

can be decomposed as follow:

$$\|X\|^2_{\widetilde{M}} = \mathrm{tr}(X^\top \widetilde{M} X) = \mathrm{diag}(\widetilde{M})^\top \mathrm{diag}(XX^\top) = \sum_v \widetilde{M}_v \|\mathrm{row}_v(X)\|^2$$

Thus we can define the metric E in Equation (2) for each vertex v:

$$E = \sum_v E_v \quad \text{and} \quad E_v = \widetilde{M}_v \|\mathrm{row}_v(PZ - \widetilde{M}^{-1}\widetilde{L}PF)\|^2.$$

We can observe that only the 1-ring of v matters, as the cost will not change for vertices further away from this region (Figure 4). More precisely, noting $\mathcal{H} = \{u,v\} \cup \mathcal{N}_1(u,v)$ when collapsing edge $e = (u,v)$, the metric only changes for vertices of $\mathcal{H}$. Therefore, we only need the 2-ring of $\{u,v\}$ to compute the cost of a given edge collapse:

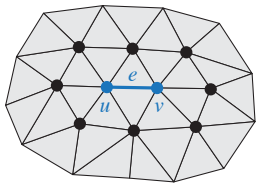


Figure 4: *When collapsing e (blue), only the 1-ring entries of E_w (black) change.*

$$\begin{aligned}\mathrm{cost}(e) &= E^{after} - E^{before} \\ &= \sum_{w\in\mathcal{H}} E_w^{after} + \sum_{w\notin\mathcal{H}} E_w^{after} - \sum_{w\in\mathcal{H}} E_w^{before} - \sum_{w\notin\mathcal{H}} E_w^{before} \\ &= \sum_{w\in\mathcal{H}} E_w^{after} - \sum_{w\in\mathcal{H}} E_w^{before}\end{aligned}$$

As a result, we only need to compute E_w for $w \in \mathcal{H}$, allowing to track the global cost via local updates. Note that the signals to preserve, F, are very important when determining which frequencies to keep during the decimation process. Consequently, we use the K first eigenvectors of L, to focus only on the low-frequencies of interest. Finally, once the edge is collapsed, with Q the restriction matrix associated to this operation, we update P, F, and Z as follow: $P \leftarrow QP$, $F \leftarrow QF$, $Z \leftarrow QZ$.

3.4. Merged vertex optimization

When collapsing an edge, we need to reposition the resulting merged vertex. This position impacts our metric via P, $\widetilde{M}$ and $\widetilde{L}$, in a non-linear fashion. Thus, we use an approximation of the metric to find its minimizer. Several strategies are possible: (i) always merge at the edge center, (ii) use a 1D quadratic approximation, restricted on the edge, or (iii) use a 3D quadratic approximation, unrestricted. Experimentally, we found that (ii) provides the best trade-off between accuracy and computational cost (see Appendix A for more details).

More precisely, let $e = (u,v)$ be the edge to collapse, the resulting position is denoted $w(\alpha \in [0,1]) = (1-\alpha)u + \alpha v$, with $\mathrm{cost}(e,\alpha)$ the collapse cost. We construct the quadratic polynomial p such that $p(\alpha) = \mathrm{cost}(e,\alpha)$ for $\alpha \in \{0, 0.5, 1\}$, and optimize $\alpha^* \in [0,1]$ to minimize p, yielding the optimal merged position (Figure 5).

The restriction matrix associated with the collapse is $Q \in \mathbb{R}^{n-1\times n}$ with n the number of vertices prior to the collapse. Let $\hat{w}$ be the index of vertex w post collapse, and v the removed vertex: the only non-zeros are $Q_{\hat{u}u} = 1-\alpha$, $Q_{\hat{u}v} = \alpha$, and $Q_{\hat{w}w} = 1$. If needed by the application, we can force Q to be binary by rounding $1-\alpha$ and α.

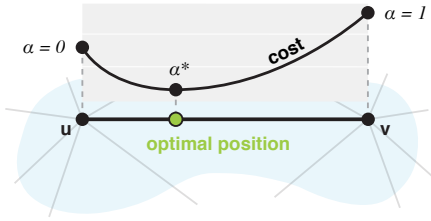


Figure 5: *Our vertex optimization scheme finds $\alpha^* \in [0,1]$ which minimizes the cost to determine the merged position (green).*

4. Evaluation

We evaluated the performance of our simplification method using a variety of criteria. We implemented our technique in C++ with the help of Spectra, and tested it on a workstation with an Intel Xeon 3.0 GHz CPU, 32 GB of RAM. We used the same dataset as Liu et al. [LJO19]. In all figures, functional maps are 100×100, and we aim to preserve 100 eigenvectors ($K = 100$).

4.1. Functional maps

First, several quantities used in this evaluation are from the functional maps field; let us briefly introduce the concept of functional maps [OBCS*12] here. These are maps between two shapes, but instead of having a point-to-point map, we use a linear mapping between function spaces. Given a base of functions on each shape $\Phi_i, \widetilde{\Phi}_j$, we can map one base function (say, Φ_i) on the base functions of the other shape (here, $\sum_j C_{ij}\widetilde{\Phi}_j$). This allows to map any function that we can decompose on these bases from one shape to the other. The functional map can be represented by the matrix $C = (C_{ij})$.

Here we consider the functional map $C \in \mathbb{R}^{K\times K}$ between the input and output shapes, so we don't take high frequencies into account. Noting Φ the matrix whose columns are the K first eigenvectors of L (and idem for $\widetilde{\Phi}$), this matrix can be defined as:

$$C = \widetilde{\Phi}^\top \widetilde{M} P \Phi$$

and given a function f on $\mathcal{M}$ ($f = \Phi x$), its corresponding function on $\widetilde{\mathcal{M}}$ is ($g = \widetilde{\Phi} C x$). To enable fair comparison with other methods, we normalize all eigenvectors: $\forall i, \|\Phi_i\|_M = 1$ and $\forall j, \|\widetilde{\Phi}_j\|_{\widetilde{M}} = 1$. We also scale the meshes to have a unit area: $\mathrm{tr}(M) = \mathrm{tr}(\widetilde{M}) = 1$.

4.2. Norms

Ideally, the functional map C between input and output should be as close as possible to the identity. In order not to rely only on visual inspection, we use two norms on the functional maps to quantify the result of the simplification:

$$\text{Laplacian commutativity: } \|C\|^2_L = \frac{\|C\Lambda - \widetilde{\Lambda}C\|^2}{\|C\|^2} \tag{3}$$

$$\text{Orthonormality: } \|C\|^2_D = \|C^\top C - \mathrm{Id}\|^2 \tag{4}$$

where Λ and $\widetilde{\Lambda}$ are the diagonal matrices of the eigenvalues of the Laplacian operator on the input and output mesh, respectively. While the ideal functional map is often the identity, multiplicity of eigenvalues may occur (e.g., in spheres or the bumpy cube in Figure 6).

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

Both norms are valid in these cases. While Equations (2) & (4) are not related, we have that if $\|C\| \neq 0$ then $E = 0 \iff \|C\|_L = 0$ (proof in Appendix B).

Our goal is to show that C is orthonormal and commutes with the Laplacians in the reduced basis if and only if it preserves corresponding eigenfunctions and eigenvalues exactly. We do not assume any constraints on C (e.g., association with a pointwise map).

Theorem 1. *For a square functional map, the following statements are equivalent:*

(1) $C^\top C = \mathrm{Id}$ *and* $C\Lambda = \widetilde{\Lambda}C$
(2) the set of functions $Y = \widetilde{\Phi}C$ *is orthonormal on* $\widetilde{\mathcal{M}}$ *and solves the eigenvalue problem* $\widetilde{L}Y = \widetilde{M}Y\widetilde{\Lambda}$ *and moreover* $\Lambda = \widetilde{\Lambda}$
(3) the set of functions $X = \Phi C^\top$ *is orthonormal on* $\mathcal{M}$ *and satisfies* $LX = MX\Lambda$ *and moreover* $\Lambda = \widetilde{\Lambda}$

Intuitively condition (2) (respectively (3)) above implies that C (respectively $C^\top$) preserves the given set of eigenpairs of the Laplacian. Then, the theorem can also be stated simply as follows: C is both orthonormal and commutes with the diagonal matrices of eigenvalues, if and only if it preserves the eigenfunctions and their corresponding eigenvalues of the Laplacians. We provide a proof of this theorem as Appendix C.

4.3. Analysis

Type of operator. The kind of Laplacian operator on the output shape differs between the compared methods. The operator retrieved by the method of Nasikun et al. [NBH18] uses geodesic distances to determine the sparsity of the Laplacian. This could yield a denser Laplacian compared to the cotangent Laplacian on a triangle mesh. The method of Liu et al. [LJO19] use the 3-ring of vertices for the Laplacian. Both methods specifically optimize for the operator.

On the contrary, our method and the method of Garland & Heckbert [GH97] output a mesh from which we can compute the Laplacian operator using standard formulations on the 1-ring; we use the usual cotangent Laplacian for the evaluations. However, this formulation is more constrained, and while usually the Laplacian operator is easily derived from the input mesh, our problem is an inverse problem: finding the best reduced mesh that respects a specific Laplacian operator (given by its eigenpairs).

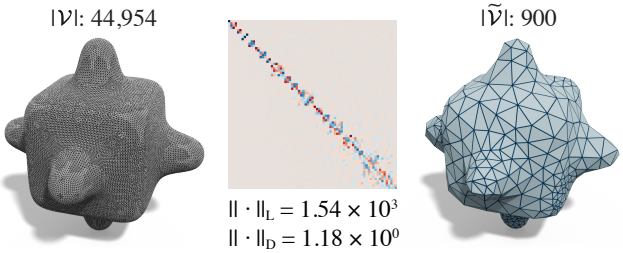


Figure 6: *The functional map from the reduction should be block-diagonal following the multiplicity of the eigenvalues.*

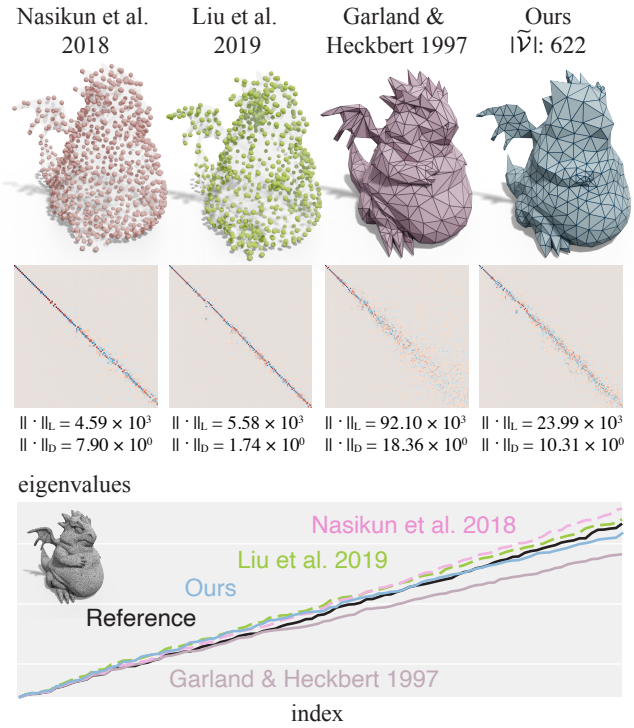


Figure 7: *To spectrally preserve the Laplacian, both coarse eigenvectors and eigenvalues should be close to their fine equivalent.*

Norms. As shown before, the ideal case is when both the Laplacian commutativity and the orthogonality are zero, as it means the spectrum is exactly preserved. We observed that in general, the method of Nasikun et al. [NBH18] and of Liu et al. [LJO19] outperform our method for small ratios of output size on input size, and the method of Nasikun et al. is still at least on par when increasing the ratio. The operator coarsening of Liu et al., however, becomes less accurate as the ratio increases, notably because more vertices means more coefficients in the operator to optimize, making the optimization more difficult. The method of Garland & Heckbert [GH97] is usually worse than our method for both metrics, and often create slivers in the output shape that will heavily impact the Laplacian operator. We show typical examples of the preservation of eigenvectors in Figure 3, of eigenvalues in Figure 7, and of the norms in Figure 8.

Storage. The method of Nasikun et al. [NBH18] aims at generating coarse eigenpairs, which are dense matrices and thus are very costly to store. Instead, as with the method of Liu et al. [LJO19], one could store only the resulting *sparse* operator, along with the selected vertices. This is still a lot larger than simply storing a coarse mesh. We measured the storage size as a function of the target size, in bytes/vertex (B/v), yielding a median of 228 B/v for the method of Nasikun et al. [NBH18], 262 B/v for the method of Liu et al. [LJO19], and 48 B/v for our method (more details in Appendix D). We require the same storage as the method of Garland & Heckbert [GH97], but with a higher quality Laplacian.

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

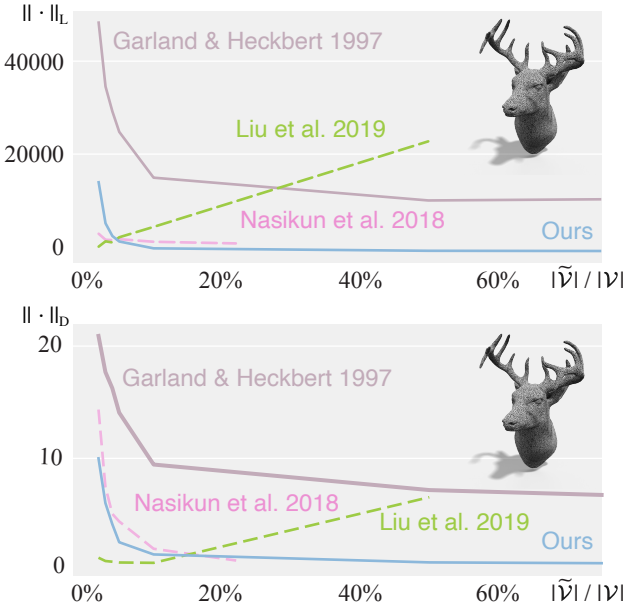


Figure 8: *One can see from these plots, $\|\cdot\|_L$ and $\|\cdot\|_D$ over the output size relative to the input size, that most methods have the similar behaviors, except for the method of Liu et al. [LJO19] for which the optimization becomes harder with more output vertices. The simplification of Garland & Heckbert [GH97] distort the spectrum and thus exhibit higher norms.*

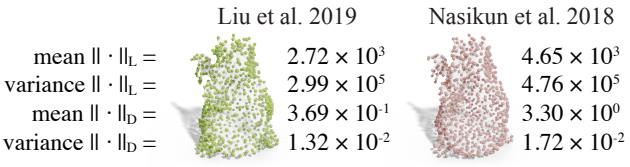


Figure 9: *Both [LJO19] and [NBH18] are not deterministic methods. Although the average of the commutativity and the orthogonality out of ten runs are small, they still have high variance.*

Determinism. Previous methods from Nasikun et al. [NBH18] and Liu et al. [LJO19] have a vertex selection step, following some regularity metric. For both of these methods, this step is initialized with a random selection, making these methods non-deterministic (Figure 9). This alters not only the final result, but also the time needed to compute the output. Mirroring Garland & Heckbert [GH97], our method is deterministic as it only depends on the input.

Timings. Similarly to the method of Liu et al. [LJO19], the initial eigenvectors computation depends heavily on the input size and can be accelerated via a faster eigen solver. Then the reduction time is linear in the number of removed vertices, as can be seen in Figure 10. This behavior is similar for the method of Garland & Heckbert [GH97], although the latter is much faster as not only the metric is defined per vertex instead of per edge, but also their setup step is less intensive. As the operator coarsening method [LJO19] optimizes for a Laplacian operator whose size depends on the output size, it

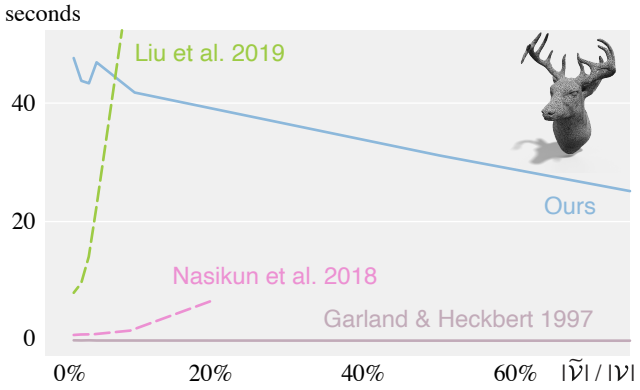


Figure 10: *Typical reduction time (seconds) in function of the ratio output size / input size. Here, $|\mathcal{V}| = 20685$, the method of Liu et al. [LJO19] was run using MATLAB, and the method of Nasikun et al. [NBH18] ran out of memory past 21% of input size.*

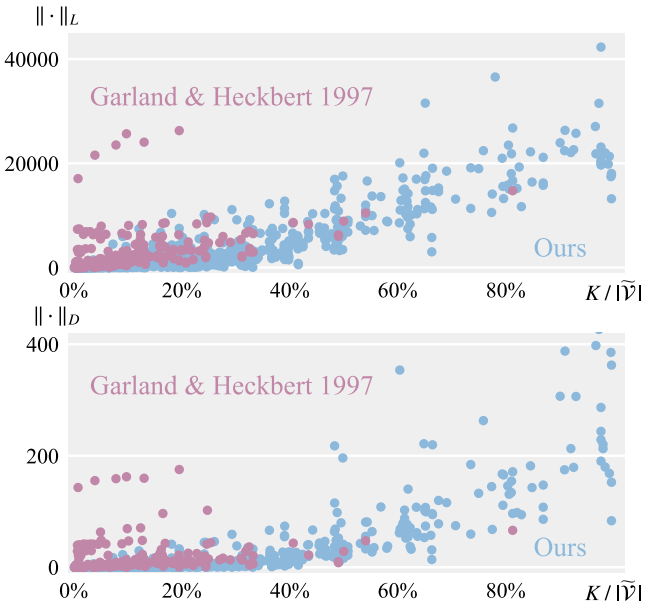


Figure 11: *By looking at the norms in function of number of eigenvectors relative to the output size, we can see that the output size should be 3 times the number of eigenvectors for a correct spectral preservation of the Laplacian.*

can be quite fast for extreme reductions but timings quickly increase following the number of vertices. Similarly, the method of Nasikun et al. [NBH18] can be really fast for small output sizes, notably due to GPU usage, and is slower when reducing less, while consuming much more memory. Although Figure 10 show the timings for the deer head, these behaviors are typical across all of our test dataset, as can be seen in Figure 20 in the appendix.

Number of eigenvectors. Increasing the number of eigenvectors in the metric of Equation (2) leads to a better preservation of high frequencies, up to a certain point. We observed from our test dataset

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

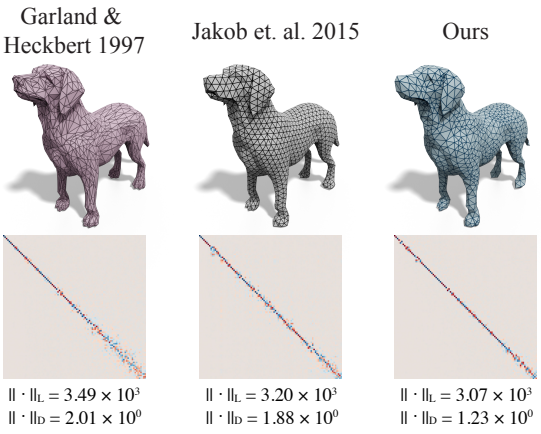


Figure 12: *Both the shape (by removing small details) and the discretization (by avoiding slivers) impact the preservation of the spectrum (here we remove 90% of the vertices).*

that the output size should be at least 3x the number of eigenvectors to preserve (Figure 11), for a correct spectral preservation of the Laplacian. From Figure 11, we also observe that Laplacian commutativity has less variability than orthogonality for high ratios.

Factors of spectral properties. Both the shape and its discretization impact the spectral properties, especially when the number of vertices is limited. Ideally, faces should be regular, to get a high quality Laplacian operator [WMKG07]. As the method of Garland & Heckbert [GH97] focuses on the shape, we also evaluated a remeshing method that focuses on the discretization: Instant Field-aligned Meshes (IFM) [JTPSH15]. On average, for the same mesh and target size, functional maps from IFM have better orthogonality while those from our method show better laplacian commutativity. However, IFM often produced non-manifold meshes, especially in presence of thin features. As with the method of Garland & Heckbert [GH97], we have the guarantee to stay manifold. When remeshing, and for a given vertex budget, it seems the spectrum is better preserved when focusing on the discretization [JTPSH15] than on the shape [GH97] (see Figure 12).

5. Applications

Our simplification method was designed to enable faster computationally expensive shape analysis tasks, by replacing dense input meshes with coarser substitutes, yet optimized to carry on as much as possible the original spectral properties onto which these task build upon. We illustrate this behavior for two applications: spectral distance computation and functional map generation.

5.1. Spectral distances

We now show the preservation of spectral distances between vertices, and evaluated several different distances. Noting ϕ_i the i-th eigenvector of the Laplacian operator on the mesh and λ_i its associated eigenvalue, these distances are detailed in the following:

$$d_{diffusion}(u,v,t) = \sum_i (\phi_i(u) - \phi_i(v))^2 e^{-2\lambda_i t}$$

$$d_{biharmonic}(u,v) = \sum_i (\phi_i(u) - \phi_i(v))^2 / \lambda_i^2$$

$$d_{WKS}(u,v) = \int_{t_{min}}^{t_{max}} \left| \frac{\mathrm{WKS}(u,t) - \mathrm{WKS}(v,t)}{\mathrm{WKS}(u,t) + \mathrm{WKS}(v,t)} \right| dt$$

$$d_{commute}(u,v) = \sum_i (\phi_i(u) - \phi_i(v))^2 / \lambda_i$$

where WKS is the wave kernel signature [ASC11]; we also evaluated the heat kernel signature (HKS) [SOG09]:

$$\mathrm{WKS}(v,t) = \sum_i \phi_i^2(v) e^{-\frac{(t-\log\lambda_i)^2}{2\sigma^2}} / \sum_i e^{-\frac{(t-\log\lambda_i)^2}{2\sigma^2}}$$

$$heat_kernel(u,v,t) = \sum_i \phi_i(u)\phi_i(v) e^{-\lambda_i t}$$

$$\mathrm{HKS}(v,t) = heat_kernel(v,v,t) = \sum_i \phi_i^2(v) e^{-\lambda_i t}$$

While heavily reducing the input mesh will decrease the quality of these distances and signatures, our method preserves them better than the method of Garland & Heckbert [GH97]. In particular, distances on meshes reduced using their method exhibit a lot more spurious local optimums than on meshes reduced with our method (see Figure 13). The Heat Kernel Signature tends to be more resistant at small values of t, but is less conserved at large values of t (see Figure 14).

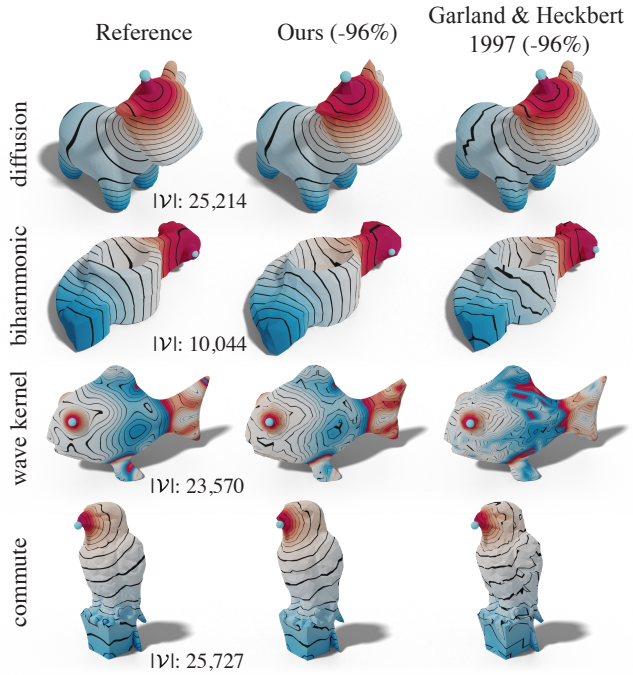


Figure 13: *Spectral distance comparison: the source point is depicted in blue and the iso-lines in black, with t set to 0.01 for the diffusion distance. With 25x less vertices in these reduced meshes, computing spectral distances is on average 18x faster.*

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

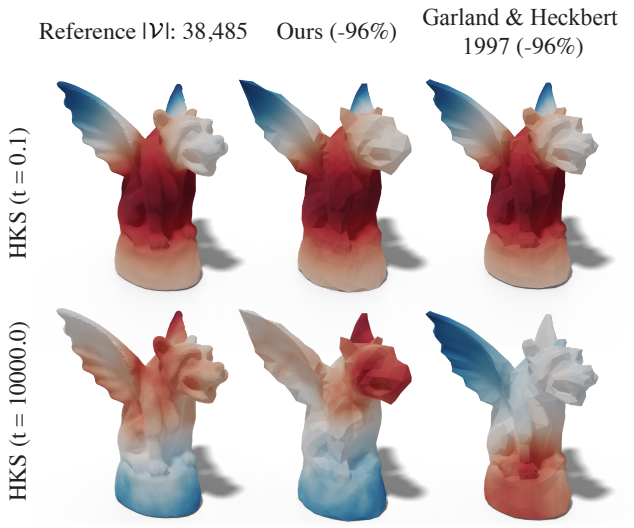


Figure 14: *Despite the high reduction, the heat kernel signature is still similar between coarse and fine mesh when using our method.*

5.2. Faster functional maps

As briefly shown before, functional maps are a powerful tool for finding correspondences between shapes, and are complemented with approaches to retrieve a point-to-point mapping. One can perform shape matching using Product Manifold Filter (PMF) [VLB*17] or Bijective and Continuous ICP (BCICP) [RPWO18], with excellent results. Both methods are iterative: PMF solves a linear assignment problem at each iteration to determine a bijection between shapes, which is has a high algorithmic complexity and also needs the shapes to have the same number of vertices. BCICP instead does not need the shapes to have an equal number of vertices, and refines both the functional map and the point-to-point maps at each iteration. However these methods do not scale performance-wise when the number of vertices increases.

This performance problem can be circumvented by simplifying the meshes prior to the matching, usually using the method of Garland & Heckbert [LRR*17], yielding a hierarchical scheme (see Figure 15). This simplification is unaware of the spectrum to preserve and can distort it, limiting the accuracy of the match. We show that we can use our method to enable robust matching, while still being faster than without the reduction. This hierarchical scheme can be written in matrix form:

$$C_{\mathcal{X},\tilde{\mathcal{Y}}} = C_{\mathcal{Y},\tilde{\mathcal{Y}}} C_{\mathcal{X},\mathcal{Y}} = C_{\tilde{\mathcal{X}},\tilde{\mathcal{Y}}} C_{\mathcal{X},\tilde{\mathcal{X}}}$$

where $C_{\mathcal{X},\mathcal{Y}}$ is the functional map from shape $\mathcal{X}$ to shape $\mathcal{Y}$. We retrieve $C_{\mathcal{X},\tilde{\mathcal{X}}}$ and $C_{\mathcal{Y},\tilde{\mathcal{Y}}}$ from our mesh simplification method, and $C_{\tilde{\mathcal{X}},\tilde{\mathcal{Y}}}$ from either PMF [VLB*17] or BCICP [RPWO18]. Then, $C_{\mathcal{X},\mathcal{Y}}$ can be computed by solving a least squares system. While the method of Liu et al. [LJO19] is also suitable for PMF but not for BCICP since the latter require a connectivity, we can use our method with both PMF and BCICP. We evaluated using BCICP in the hierarchical functional maps, and compare the results to a reduction with the method of Garland & Heckbert [GH97]. As can be observed in Figure 16, our method enable better matching, and is significantly faster than running BCICP on the fine shape. Indeed, while running BCICP on meshes with 600 vertices to 1 minute *per shape pair*, BCICP on meshes with 1000 vertices took on average 2 minutes, and ran out of memory for mesh around 30K vertices. For this application, we used binary restriction matrices.

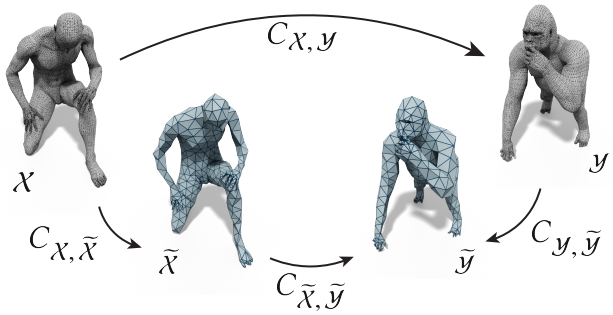


Figure 15: *One can accelerate the computation of functional maps between detailed meshes by performing shape matching on simplified shapes before upscaling the resulting functional maps.*

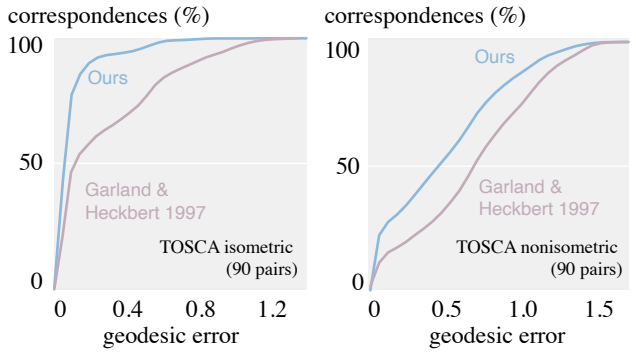


Figure 16: *Reducing meshes from the TOSCA dataset from around 30K vertices to 600 vertices allowed to use BCICP [RPWO18] as it would run out of memory on the fine meshes, and our method provides better correspondences than the simplification of Garland & Heckbert [GH97]. On the reduced meshes, the BCICP computation took on average 67s, from 16s to 142s (variance: 802).*

6. Discussion

Edge flips. The cost of an edge collapse and the preservation metric are defined in such a way that we can easily extend it to edge flips, keeping only edge flips with a negative cost in order to avoid cycles (Figure 17). We tried this during the reduction process, but this always overfitted and generated poor results (Figure 18). Doing a post-process optimization using edge flips does indeed lead to a

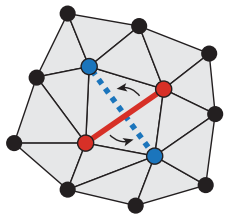


Figure 17: *An edge flip will not change the E_w corresponding to the 1-ring vertices (black) and beyond.*

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

result of better quality, but this is quantifiably marginal and is often not worth the time and complexity.

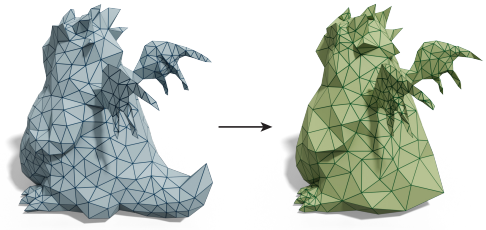


Figure 18: *Allowing edge flips often results in whole parts missing.*

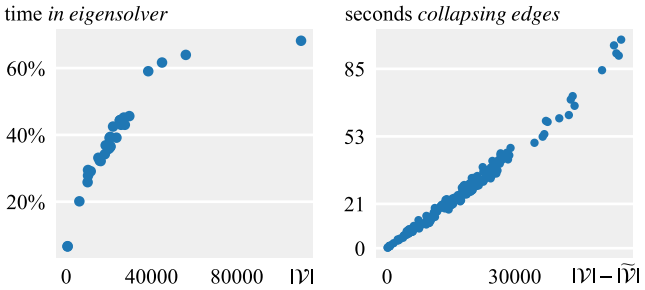


Figure 19: *For large meshes, most of the time is spent in the eigensolver (left, with $|\widetilde{\mathcal{V}}| = 10\%|\mathcal{V}|$). After the setup step, the reduction time (right) is linear in the number of collapsed edges.*

Limitations. There are at least two main areas where our approach can further be developed, both related to time performance. First, the eigenvectors need to be computed at the beginning of the algorithm, while one may seek computing them at query time, when needed. Second, our cost evaluation involves fairly large matrices, at each evaluation which could be optimized using an approximation scheme.

Scalability. By far, the most limiting factor when scaling to large meshes is the eigen solver, which can be seamlessly replaced by a faster one (see Figure 19). Interestingly, one could use a fast approximation [NBH18] for this step. Lowering the number of eigenvectors K would help for both the setup and the reduction, as the matrices to manipulate would be smaller. It is also possible to partially reduce the input using the QEM [GH97] (e.g., down to 100K or 200K vertices) before using our method: the impact of such pre-processing on the spectrum would be limited.

7. Conclusion

We have introduced a new mesh simplification algorithm which is designed to preserve, as much possible, the spectral properties of the input surface. Our method is built on a standard graph reduction algorithm for which we introduced a custom metric driving a cost designed specifically to preserve the spectrum, together with a repositioning strategy for merged vertices. We illustrated the superior behavior of our decimation scheme compared to appearance preserving methods, for spectral distance computation and functional map generation. Yet, we believe more spectrum-dependent applications may find immediate benefit from our approach.

Acknowledgments

Parts of this work were supported by the KAUST OSR Award No. CRG-2017-3426, the ERC Starting Grant No. 758800 (EXPROTEA), NSERC Discovery (RGPIN2017–05235, RGPAS–2017–507938), the Ontario Early Research Award program, the Canada Research Chairs Program, the Fields Centre for Quantitative Analysis and Modelling and gifts by Adobe Systems, Autodesk, and MESH Inc.

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

References

[ASC11] AUBRY M., SCHLICKEWEI U., CREMERS D.: The wave kernel signature: A quantum mechanical approach to shape analysis. In *Proc. ICCV* (2011). 7

[CBO*19] CHEN J., BUDNINSKIY M., OWHADI H., BAO H., HUANG J., DESBRUN M.: Material-adapted refinable basis functions for elasticity simulation. *ACM Trans. on Graphics* (2019). 2

[CMO97] COHEN J., MANOCHA D., OLANO M.: Simplifying polygonal models using successive mappings. In *Proc. IEEE Vis* (1997). 2

[CS18] CLAICI S., SOLOMON J.: Wasserstein coresets for Lipschitz costs. *arXiv preprint arXiv:1805.07412* (2018). 2

[CSAD04] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. *ACM Trans. on Graphics* (2004). 2

[CVM*96] COHEN J., VARSHNEY A., MANOCHA D., TURK G., WEBER H., AGARWAL P., BROOKS F., WRIGHT W.: Simplification envelopes. In *Proc. SIGGRAPH* (1996). 2

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *Proc. SIGGRAPH* (1997). 1, 2, 3, 5, 6, 7, 8, 9

[GH98] GARLAND M., HECKBERT P. S.: Simplifying surfaces with color and texture using quadric error metrics. In *Proc. IEEE Vis* (1998). 2

[HCB16] HUGGINS J., CAMPBELL T., BRODERICK T.: Coresets for scalable Bayesian logistic regression. In *Proc. NeurIPS* (2016). 2

[HDD*93] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Mesh optimization. In *Proc. SIGGRAPH* (1993). 2

[Hop99] HOPPE H.: New quadric metric for simplifying meshes with appearance attributes. In *Proc. IEEE Vis* (1999). 2

[JTPSH15] JAKOB W., TARINI M., PANOZZO D., SORKINE-HORNUNG O.: Instant field-aligned meshes. *ACM Trans. on Graphics* (2015). 7

[KS16] KYNG R., SACHDEVA S.: Approximate Gaussian elimination for Laplacians-fast, sparse, and simple. In *Proc. FOCS* (2016). 2

[LFJG17] LIU S., FERGUSON Z., JACOBSON A., GINGOLD Y. I.: Seamless: seam erasure and seam-aware decoupling of shape from mesh resolution. *ACM Trans. on Graphics* (2017). 2

[LFZ15] LI D., FEI Y., ZHENG C.: Interactive acoustic transfer approximation for modal sound. *ACM Trans. on Graphics* (2015). 2

[LJO19] LIU H.-T. D., JACOBSON A., OVSJANIKOV M.: Spectral coarsening of geometric operators. *ACM Trans. on Graphics* (2019). 2, 3, 4, 5, 6, 8, 10

[Lou19] LOUKAS A.: Graph reduction with spectral and cut guarantees. *J. Machine Learning Research* (2019). 2

[LRR*17] LITANY O., REMEZ T., RODOLÀ E., BRONSTEIN A. M., BRONSTEIN M. M.: Deep functional maps: Structured prediction for dense shape correspondence. In *Proc. ICCV* (2017). 8

[LV18] LOUKAS A., VANDERGHEYNST P.: Spectrally approximating large graphs with smaller graphs. In *Proc. ICML* (2018). 2

[NBH18] NASIKUN A., BRANDT C., HILDEBRANDT K.: Fast approximation of Laplace-Beltrami eigenproblems. *Comp. Graph. Forum* (2018). 2, 5, 6, 9

[OBCS*12] OVSJANIKOV M., BEN-CHEN M., SOLOMON J., BUTSCHER A., GUIBAS L.: Functional maps: A flexible representation of maps between shapes. *ACM Trans. on Graphics* (2012). 1, 2, 4

[OS19] OWHADI H., SCOVEL C.: *Operator-Adapted Wavelets, Fast Solvers, and Numerical Homogenization: From a Game Theoretic Approach to Numerical Approximation and Algorithm Design*. 2019. 2

[PH97] POPOVIĆ J., HOPPE H.: Progressive simplicial complexes. In *Proc. SIGGRAPH* (1997). 2

[RPWO18] REN J., POULENARD A., WONKA P., OVSJANIKOV M.: Continuous and orientation-preserving correspondences via functional maps. *ACM Trans. on Graphics* (2018). 8

[SGG*00] SANDER P. V., GU X., GORTLER S. J., HOPPE H., SNYDER J.: Silhouette clipping. In *Proc. SIGGRAPH* (2000). 2

[SOG09] SUN J., OVSJANIKOV M., GUIBAS L.: A concise and provably informative multi-scale signature based on heat diffusion. In *Proc. SGP* (2009). 7

[SVJ15] SACHT L., VOUGA E., JACOBSON A.: Nested cages. *ACM Trans. on Graphics* (2015). 2

[SZL*92] SCHROEDER W. J., ZARGE J. A., LORENSEN W. E., ET AL.: Decimation of triangle meshes. In *Proc. SIGGRAPH* (1992). 2

[VLB*17] VESTNER M., LÄHNER Z., BOYARSKI A., LITANY O., SLOSSBERG R., REMEZ T., RODOLA E., BRONSTEIN A., BRONSTEIN M., KIMMEL R., CREMERS D.: Efficient deformable shape correspondence via kernel matching. 8

[WMKG07] WARDETZKY M., MATHUR S., KAELBERER F., GRINSPUN E.: Discrete Laplace operators: No free lunch. In *Proc. SGP* (2007). 7

[ZG02] ZELINKA S., GARLAND M.: Permission grids: Practical, error-bounded simplification. *ACM Trans. on Graphics* (2002). 2

Appendix

A. Merged position

Strategy	$\|\cdot\|_L$	$\|\cdot\|_D$	Time
(i) middle	1.0	1.0	**1.0**
(ii) on edge	**0.7**	**0.6**	2.1
(iii) unrestricted	0.8	0.9	4.9

Table 1: *Median norms and time relative to strategy (i), for the same mesh and parameters.*

Let $\text{cost}(e,x)$ be the cost of collapsing e with x the merged position. As α is needed for P (and thus impacts the cost), we define it as $\frac{(x-u)\cdot(v-u)}{\|v-u\|^2}$ and clamp it in $[0,1]$. For strategy (iii), we sample $\text{cost}(e,x)$ for x in a sphere around e to determine the approximating quadric (in $\mathbb{R}^{4\times4}$), which we then minimize to get x^* (and α^*). We refer to Table 1 for a comparison relative to strategy (i).

B. Relation between Equations (2) & (3)

Proof. Having $E=0$ (from Equation (2)) is equivalent to $\|C\|_L=0$ (from Equation (3)). Since all coefficients of $\widetilde{M}$ are strictly positive, $E=0 \iff PM^{-1}L\Phi = \widetilde{M}^{-1}\widetilde{L}P\Phi$. As $L\Phi = M\Phi\Lambda$, the left-hand side is equal to $P\Phi\Lambda$. We left-multiply both sides by $\widetilde{\Phi}^\top\widetilde{M}$, yielding $C\Lambda = \widetilde{\Phi}^\top\widetilde{L}P\Phi$ since $C=\widetilde{\Phi}^\top\widetilde{M}P\Phi$. Recalling that $\widetilde{L}\widetilde{\Phi}=\widetilde{M}\widetilde{\Phi}\widetilde{\Lambda}$, the right-hand side is equal to $(\widetilde{M}\widetilde{\Phi}\widetilde{\Lambda})^\top P\Phi$, thus $\widetilde{\Lambda}C$. This means that $C\Lambda=\widetilde{\Lambda}C$, proving that $E=0 \iff \|C\|_L=0$. □

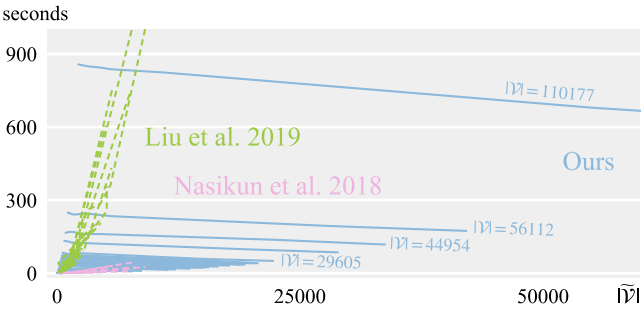


Figure 20: *Timings as a function of $|\widetilde{\mathcal{V}}|$ with $|\mathcal{V}|$ fixed per curve. The behavior shown in Figure 10 is also visible here.*

C. Proof of Theorem 1

Proof. We will prove the equivalence between (1) and (2). The equivalence between (1) and (3) is proved identically. Suppose (2) holds. To show that (1) must hold, first note that from orthonormality of Y on $\widetilde{\mathcal{M}}$ by definition we get $Y^\top\widetilde{M}Y=\text{Id}$, i.e. $C^\top\widetilde{\Phi}^\top\widetilde{M}\widetilde{\Phi}C=\text{Id}$, and since $\widetilde{\Phi}$ is orthonormal on $\widetilde{\mathcal{M}}$ this implies $C^\top C=\text{Id}$. Moreover, by assumption $\widetilde{L}Y=\widetilde{M}Y\widetilde{\Lambda}$, and $\Lambda=\widetilde{\Lambda}$. Thus, $\widetilde{L}\widetilde{\Phi}C=\widetilde{M}\widetilde{\Phi}C\Lambda$. Since by definition $\widetilde{L}\widetilde{\Phi}=\widetilde{M}\widetilde{\Phi}\widetilde{\Lambda}$ we get $\widetilde{M}\widetilde{\Phi}\widetilde{\Lambda}C=\widetilde{M}\widetilde{\Phi}C\Lambda$ which implies $\widetilde{\Lambda}C=C\Lambda$.

Conversely, suppose that (1) holds. If $Y=\widetilde{\Phi}C$, then $C^\top C=\text{Id}$ implies: $Y^\top\widetilde{M}Y=C^\top\widetilde{\Phi}^\top\widetilde{M}\widetilde{\Phi}C=\text{Id}$, since $\widetilde{\Phi}$ is orthonormal with respect to $\widetilde{M}$. Now, $\widetilde{L}Y=\widetilde{L}\widetilde{\Phi}C=\widetilde{M}\widetilde{\Phi}\widetilde{\Lambda}C$. Since $\widetilde{\Lambda}C=C\Lambda$ by assumption, we get $\widetilde{L}Y=\widetilde{M}\widetilde{\Phi}C\Lambda=\widetilde{M}Y\Lambda$. Therefore, Y solves the eigenvalue problem of $(\widetilde{L},\widetilde{M})$ with the eigenvalues Λ. It remains to prove that $\Lambda=\widetilde{\Lambda}$. For this, note that $\widetilde{\Lambda}C=C\Lambda$ implies $C_{ij}^2(\widetilde{\lambda}_i-\lambda_j)^2=0\ \forall i,j$, where $\widetilde{\lambda}_i$ is the i^{th} eigenvalue of $\widetilde{L}$ (idem for L). Using this and $C^\top C=\text{Id}$ implies that C must be block orthonormal with blocks corresponding to the equal eigenvalues (to see this, note that for each $\widetilde{\lambda}_i$ there must be an equal λ_j otherwise a row or column of C would have to be zero). Moreover the blocks must be square by orthonormality of C, so that $\Lambda=\widetilde{\Lambda}$. □

D. Storage sizes

Let n be the target size and c the number of coefficient per row for $\widetilde{L}$. We only consider 64bit floats here. We can store the selected subset of vertices via a list of n 32bit indices. $\widetilde{M}$, which is diagonal, requires n floats. $\widetilde{L}$ is symmetric so we need to store only $n(c+1)/2$ coefficients, each one taking one float and two 16bit integers ($\widetilde{L}$ never exceeds 65,535 rows in our tests). Overall, this gives $(18+6c)n$ bytes. Considering the observed median size for the method of Liu et al. [LJO19] (about 262 B/v), we find $c\approx41$, close to what the authors report just before their Section 3.2.

For meshes, we need $3n$ floats for vertices. Assuming an average valence of 6, there are $2n$ triangles, needing $3\times2n$ 32bit integers. This yields a total of $48n$ bytes, which we experimentally observe.

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.