Fast Approximation of Laplace-Beltrami Eigenproblems

Nasikun, Ahmad; Brandt, Christopher; Hildebrandt, Klaus

Publication date
2018

Document Version
Final published version

Published in
Computer Graphics Forum

Citation (APA)
Nasikun, A., Brandt, C., & Hildebrandt, K. (2018). Fast Approximation of Laplace-Beltrami Eigenproblems. *Computer Graphics Forum*, *37*(5), 121-134.

DOI: 10.1111/cgf.13496
Eurographics Symposium on Geometry Processing 2018
T. Ju and A. Vaxman
(Guest Editors)

Volume 37 (2018), Number 5

Fast Approximation of Laplace–Beltrami Eigenproblems

Ahmad Nasikun Christopher Brandt Klaus Hildebrandt

Delft University of Technology

Abstract

The spectrum and eigenfunctions of the Laplace-Beltrami operator are at the heart of effective schemes for a variety of problems in geometry processing. A burden attached to these spectral methods is that they need to numerically solve a large-scale eigenvalue problem, which results in costly precomputation. In this paper, we address this problem by proposing a fast approximation algorithm for the lowest part of the spectrum of the Laplace–Beltrami operator. Our experiments indicate that the resulting spectra well-approximate reference spectra, which are computed with state-of-the-art eigensolvers. Moreover, we demonstrate that for different applications that comparable results are produced with the approximate and the reference spectra and eigenfunctions. The benefits of the proposed algorithm are that the cost for computing the approximate spectra is just a fraction of the cost required for numerically solving the eigenvalue problems, the storage requirements are reduced and evaluation times are lower. Our approach can help to substantially reduce the computational burden attached to spectral methods for geometry processing.

1. Introduction

The spectrum and eigenfunctions of the Laplace–Beltrami operator proved to be an effective tool for a variety of tasks in geometry processing, leading to their own research branch, called *spectral mesh processing*. Spectral methods profit from properties of the Laplace–Beltrami operator and its spectrum. From a signal processing point of view, functions on a surface can be seen as signals and the eigenbasis of the Laplace–Beltrami allows us to associate a frequency spectrum to a function analogous to the Fourier decomposition. This enables applications such as spectral filtering of functions on a mesh as well as the embedding of the mesh. Another important property of the Laplace–Beltrami operator is that it is invariant under isometric deformations of the surfaces. This property makes the spectrum attractive as an ingredient to pose-invariant shape descriptors and the eigenfunctions a tool for establishing correspondences, explicit or functional, between shapes in different poses. The downside of spectral methods on meshes is that the lowest part of the spectrum (typically the first 100-5000 eigenpairs) has to be computed. Since closed-form solutions are not available and large-scale sparse eigenproblems needs to be numerically solved, which leads to long precomputation times before spectral processing tools can be applied.

In this paper, we introduce a fast approximation algorithm for the lowest part of the spectrum and the corresponding eigenfunctions of the Laplace–Beltrami operator on surface meshes. Computing the approximation requires only a fraction of the time required for solving the original problem. For example, in our experiments, the lowest 2500 eigenvalues and eigenfunctions of a mesh with 240k vertices are approximated in less than one minute, while solving the full-resolution eigenproblem requires almost three hours. Further benefits are that the storage requirements are reduced, which enables working with larger bases in-core. Also, the computation of the approximate spectra does not require solving a large-scale eigenvalue problem but only a low-dimensional eigenproblem, which can be done by dense eigensolvers. Our experiments demonstrate that the approximated spectra are close to the reference solutions. We show that for spectral methods, such as shape DNA, diffusion distance, and spectral filtering, using the approximated spectra and eigenfunctions leads to results that closely approximate results produced with reference spectra and eigenfunctions, while requiring about two orders of magnitude shorter precomputations. The proposed approach can be applied for the computation of approximate spectra and eigenfunctions for other discrete operators as well. We show that for parameter-dependent operators, which are used for spectral shape analysis, approximations of the lowest 100 eigenvalues and eigenfunctions can be computed at interactive rates. This enables interactive exploration of the parameter space. Extending the range of applications, we apply the proposed scheme to the computation of approximations of vibration modes of elastic objects and show that our method reduces precomputation times, storage requirements and enables simulation with larger modal bases.

The idea underlying our approach is to take advantage of the fact that we can explicitly construct subspaces of the space of all functions on a mesh that include the low-frequency functions. The lowest part of the spectrum and the corresponding eigenfunctions can

be characterized as the minimizers of the Dirichlet energy subject to unit L^2-norm and pairwise L^2-orthogonality constraints. The approximation algorithm first constructs a subspace, and then solves the optimization problem restricted to the subspace. For this approach to be effective, the subspace construction needs to be fast, the subspaces should contain approximations of the low-frequency functions, and an efficient solver for the restricted optimization problem is needed. The subspace construction we propose draws on ideas used for generating weights for character skinning and shape deformation and is designed to allow for the fast construction of larger, e.g. 10k-dimensional, subspaces. Furthermore, the subspace basis is designed to be sparse, which reduces the computational cost for setting up the restricted optimization problem and allows to efficiently store and access the approximate eigenbasis and the subspace matrix. Since the approximate eigenpairs are minimizers of the restricted optimization problems, they also preserve properties of the true eigenfunction, e.g., they form an L^2-orthonormal system in the space of functions on the mesh. For solving the restricted eigenvalue problem, we found that GPU-based dense QR solvers allow to compute all eigenpairs of the restricted problem in a reasonable time. The fact that all eigenpairs are computed helps to avoid missing eigenfunctions in eigenspace of dimension two or higher as well as eigenspaces with almost identical eigenvalues.

2. Related Work

Spectral mesh processing In the following, we discuss some spectral mesh processing methods. For an introduction to the topic, we refer to [ZvKD10]. Vallet and Lévy [VL08] explored schemes for the numerical computation of the eigendecomposition of discrete Laplace-Beltrami operators on triangle meshes. They also proposed a framework for spectral filtering of functions on a mesh. The filtering can be used to process the embedding of the surface itself which allows for surface smoothing and sharpening filters. Karni and Gotsman [KG00] introduced a method for the compression of the vertex positions of a mesh using the eigenfunctions of a combinatorial Laplace matrix. The scheme was extended to the compression for mesh sequences by Váša et al. [VMHB14]. Dong et al. [DBG*06] used the critical points of low-frequency eigenfunctions as a starting point for the construction of coarse quadrangulations of surfaces. This approach was extended to provide users with control over shape, sizes and alignment of the quadrilaterals by Huang et al. [HZM*08] and Ling et al. [LHJ*14]. Sharma et al. [SHKvL09] and Huang et al. [HWAG09] proposed spectral methods for surface segmentation. Musialski et al. [MAB*15] used the low-frequency eigenfunctions to create a low-dimensional space that describes surface deformation in order to obtain a reduced-order model for shape optimization problems. Song et al. [SLMR14] defined a saliency measure on surfaces that combined spectral and spatial information and takes advantage of the global nature of information embedded in the low-frequency eigenfunction. Spectral methods have also been used for tangential vector and n-vector field processing on surfaces [ABCCO13, AOCBC15, BSEH17, BSEH18].

Spectral shape analysis The isometry invariance and the underlying continuous formulation make the Laplace–Beltrami spectrum and eigenfunctions well-suited as a basis for mesh-invariant and pose-invariant shape descriptors and signatures. Examples of such descriptors are the *Shape-DNA* [RWP05, RWP06], the *diffusion distance* [NLCK05], the *global point signature* [Rus07], the *heat kernel signature* [SOG09], the *Auto Diffusion Function* [GBAL09] and the *wave kernel signature* [ASC11]. These shape descriptors can be combined to form bags of features that can be used to design algorithms for pose and mesh invariant shape search and retrieval [BBGO11]. In addition to their use as shape descriptors, the invariance properties of the eigenfunctions make them well-suited for the construction of shape correspondences. Ovsjanikov at el. [OBCS*12] use the eigenfunctions of the Laplace–Beltrami operator of two near-isometric shapes to construct a *functional map*, which is a linear operator between the function space of the surfaces. The functional map can be used to map information, given in the form of a function on the surface, from one surface to the other. Rustamov et al. [ROA*13] use functional maps between surface to analyzed difference between shapes. While functional map compute the eigenfunctions on the two shape independently, Kovnatsky et al. [KBB*13] propose an approach that couples the computation of the eigenfunctions of a pair of shapes using landmarks correspondences as input. Spectral methods for shape matching can profit from looking at local shape matches and partial correspondences [RCB*17, LRBB17]. For an introduction to functional maps, we refer to [OCB*16]. Spectral methods are also used in the context of *geometric deep learning* [BZSL14, BMM*15, DBV16, LMBB17, BBL*17].

Beyond Laplacian While for some applications, invariance under isometric deformations is desirable, in other settings, extrinsic information about shapes, such as sharp bends, needs to be considered. In [HSvTP10, HSvTP12b], a modified Laplace–Beltrami operator that includes information about the extrinsic curvatures of the surface is proposed. Recently, alternative constructions of extrinsic operators based on the Dirac operator [LJC17] and the Steklov eigenvalue problem [WBPS17] have been introduced. Choukroun et al. [CSBK16, CPK17] explored the construction of Schrödinger operators for spectral processing and analysis. The Schrödinger operator augments the Laplacian with a potential, which can be specifically designed for the different applications. A related construction is introduced by Melzi et al. [MRCB18]. Though we focus the presentation on the Laplace–Beltrami operator, our our approach can be used for fast approximations of the spectra and eigenfunctions for these operators as well. In Section 5, we show how the fast responses of the approximation algorithm can be used to interactively explore parameter values.

Nyström method An alternative approach for approximating the spectrum and eigenfunctions of linear operators is the *Nyström method* [WS01]. The Nyström method is used in the context of machine learning for accelerating kernel methods [DM05] and spectral clustering [FBCM04] as well as for approximating large scale singular value decompositions for manifold learning [TKMR13]. The Nyström method constructs a submatrix A of the large matrix M that is built by selecting a some landmark indices and removing all rows and columns that are not landmarks from M. An eigendecomposition of A is computed and lifted to the high-dimensional space. While this approach works well for the matrices that appear in learning applications, such as covariance matrices, it cannot be

used for the extremely sparse matrices we consider in this work. The reason is that the small matrix A constructed from matrices like the cotangent matrix is a diagonal matrix if the landmarks are not chosen to be neighboring vertices. Hence an eigendecomposition of the small matrix is trivial and does not provide additional information unless the sampling is so dense that for every landmark some neighboring vertices are also landmarks. The same holds for variants of the Nyström method, like column sampling, which selects columns of the large matrix and performs an SVD in the resulting rectangular matrix. Unless the sampling is very dense, sampling columns from the cotangent matrix results in a rectangular matrix that has only one entry per row.

Subspace projection A second method for approximating eigenproblems in machine learning is *random projection* [HMT11]. First, a random rectangular matrix A of size nm, where n is the number of variables and m the number of desired eigenvectors, is constructed. Then the large matrix M is multiplied with A one or more times, similar to power iterations. Finally a singular value decomposition of the result is computed to get approximate eigenvectors. Random projection is used to approximate the eigenvectors corresponding to the largest eigenvalues, *e.g.*, for principle component analysis. Here, we are interested in the lowest eigenvectors of matrices. To use random projection for our purposes, we would need to multiply A with the inverse of M to A towards to lowest eigenvectors. The subspace iteration method [Bat13], used in continuum mechanics for the computation of vibration modes, alternates between inverse iteration and orthonormalization for computing the lowest eigenvectors. Compared to the computational cost of our approximation algorithm, subspace projection iterations are expensive. Even a single iteration of subspace projection is far more expensive than our whole approximation algorithm.

Kernel approximations The eigenvalues and eigenfunctions of the Laplace–Beltrami operator can be used to compute the heat kernel and spectral distance. Then, using only the lowest part of the spectrum and corresponding eigenfunctions, the kernel and the distance measures can be approximated. Once the eigenproblem is solved, the kernel and the distances can be evaluated at low computational cost. Since the computation of the eigenfunctions is costly, alternative approaches for approximating the heat kernel and the spectral distances have been proposed. Vaxman et al. [VBCG10] proposed a multi-resolution hierarchy for the approximation of the heat kernel and used the scheme for diffusion-based feature extraction from surfaces. Patané [Pat17] proposed a scheme that can approximate the heat kernel and spectral distances by solving sparse linear systems. This approach reduces the precomputation time since it avoids solving an eigenvalue problem. On the other hand, compared to spectral methods, the computational cost for solving individual distance queries is higher.

3. Background: Laplace–Beltrami Eigenproblem

In this section, we first briefly introduce the continuous eigenproblem of the Laplace–Beltrami operator, then we describe the discrete setting and the discrete eigenproblem.

Continuous eigenproblem We consider a smooth, compact surface Σ and the two biliner forms

$$\langle f, g\rangle_{L^2} = \int_\Sigma f\, g \,\mathrm{d}A \tag{1}$$

and

$$\langle f, g\rangle_{H^1_0} = \int_\Sigma \langle \operatorname{grad} f, \operatorname{grad} g\rangle_\Sigma \,\mathrm{d}A \tag{2}$$

that are defined on the space H^1 of functions on Σ whose weak derivatives are square integrable. The eigenvalue problem of the Laplace–Beltrami operator is to find pairs $(\lambda, \phi) \in \mathbb{R} \times H^1$ such that

$$\langle \phi, f\rangle_{H^1_0} = \lambda \langle \phi, f\rangle_{L^2} \tag{3}$$

holds for all $f \in H^1$. Since $\langle f, f\rangle_{H^1_0}$ vanishes for constant functions f, the constant functions form a one-dimensional eigenspace with eigenvalue 0. The first non-zero eigenvalue can be characterized as the minimum of $\langle \phi, \phi\rangle_{H^1_0}$ among all functions in H^1 that have unit L^2-norm and are L^2-orthogonal to the constant functions. A similar variational characterization can be formulated for the other eigenvalues by adding the constraints, that the eigenfunctions need to be L^2-orthogonal not only to the constant functions but to all eigenfunctions with smaller eigenvalue.

Discrete setting In the discrete setting, we consider triangle meshes in $\mathbb{R}^3$ and the space of a functions that are continuous on the whole surface and linear polynomials over the triangles. The functions can be described by nodal vectors, that is, by vectors listing the function values at the vertices of the mesh. The polynomial corresponding to a nodal vector can be explicitly constructed since there is a unique linear polynomial over a triangle that interpolates three given function values at the vertices. We denote by φ_i the function that takes the value one at the i^{th} vertex and zero for all other vertices. For the continuous and piecewise polynomial functions the bilinear forms are well-defined, hence they can be represented by matrices, w.r.t. to the nodal basis. The resulting matrices M and S with entries

$$M_{ij} = \langle \varphi_i, \varphi_j\rangle_{L^2} \qquad \text{and} \qquad S_{ij} = \langle \varphi_i, \varphi_j\rangle_{H^1_0}. \tag{4}$$

are called the mass matrix and the stiffness matrix (or cotangens matrix). Explicit formulas for M_{ij} and S_{ij} can be found, for example, in [WBH*07] and [VL08].

While in our experiments we consider the setting described above, our approach can be applied to other settings, such as discrete Laplacians for polygonal meshes [AW11], higher-order finite elements on meshes [RWP05, RWP06] or Discrete Exterior Calculus discretizations [CdGDS13] as well.

Discrete eigenproblem Our goal is to compute approximations of the lowest m eigenvectors and eigenvalues of the discrete Laplace–Beltrami operator of a mesh with n vertices. Analogous to the continuous case, the m lowest eigenpairs can be characterized as solutions of the variational problem:

$$\min_{\Phi \in \mathbb{R}^{n\times m}} \operatorname{tr}\left(\Phi^T S \Phi\right) \quad \text{subject to } \Phi^T M \Phi = Id. \tag{5}$$

The columns of the minimizer Φ are the nodal vectors of the eigenfunctions and the corresponding eigenvalues are given by $\lambda_i = \Phi_i^T S \Phi_i$, where Φ_i is the i^{th} column of the minimizer Φ. The eigenpairs (λ_i, Φ_i) satisfy the equation

$$S\Phi_i = \lambda_i M \Phi_i, \tag{6}$$

which is the discrete analog of (3).

4. Fast Approximation Algorithm

In this section, we introduce our approach for the fast approximation of the lowest eigenvalues and corresponding eigenfunctions of the discrete Laplace–Beltrami operator. The idea is to construct a subspace of the space of all functions on the mesh. Then we restrict the computation of the eigenvalues and eigenfunctions to the subspace, that is, we solve the optimization problem (5) restricted to the subspace. For the approach to be effective, the subspace construction needs to be fast and the constructed subspace needs to be able to approximate eigenfunctions from the lower end of the spectrum well.

Subspace construction We construct a d-dimensional subspace of the space of continuous, piecewise linear functions on the mesh. The basis vectors that span the subspace are stored as the columns of a matrix $U \in \mathbb{R}^{n \times d}$. The construction draws on ideas used for the construction of weights spaces for skinning and deformation, such as bounded biharmonic weights [JBPS11] and the linear subspace construction proposed in [WJBK15]. However, there are essential differences to these approaches. For example, while [JBPS11] solve a box constraint quadratic optimization and [WJBK15] solves a linear system for every basis vector, we only query a local neighborhood of a sample point to the construct a basis vector. This is important for our construction since we want to be able to construct larger, e.g. $10k$-dimensional, subspaces in a few seconds.

The subspace construction proceeds in three steps. First, a point sampling of the surface is constructed. Then, basis functions, which are locally supported around the sample points, are constructed. Finally, the functions are modified to form a partition of unity.

The goal of the sampling stage is to select a subset of the set of vertices from the mesh such that the subset provides an evenly distributed sampling of the surface. A requirement for the choice of the sampling scheme is that it needs to be able to compute a sampling with several thousand sample points in a few seconds. In our experiments, we observed that the constrained Poisson-disk sampling on triangle meshes introduced in [CCS12] satisfies our requirements. We denote the indices of the sample vertices by $\{s_1, s_2, ..., s_d\}$ and the set of sampled vertices by $\{v_{s_1}, v_{s_2}, .., v_{s_d}\}$. Examples of samplings are shown in Figure 1.

In the second step, we construct a preliminary matrix $\tilde{U} \in \mathbb{R}^{n \times d}$. The i^{th} column of the matrix represents a locally supported function centered at the sample point v_{s_i}. The function takes the value one at v_{s_i}, monotonically decreases (in radial direction) in a neighborhood around v_{s_i}, and vanishes outside of the neighborhood. The size of the support of the functions is controlled by a global parameter ρ . We denote by $d(v_i, v_{s_j})$ the geodesic distance between v_i and v_{s_j}

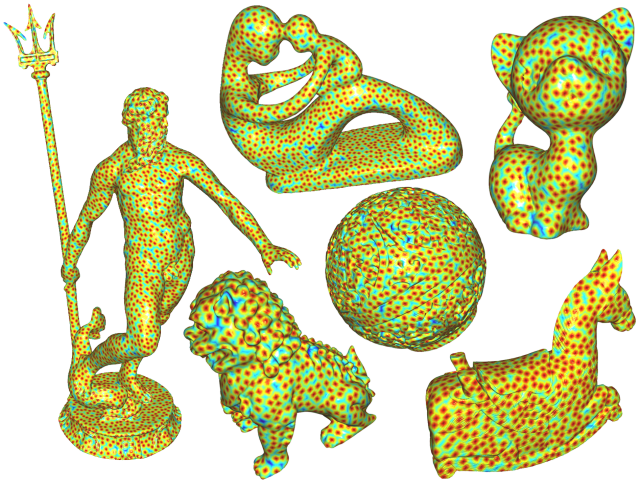

Figure 1: *Examples of samplings used for the construction of the subspace bases showing between 1000 and 5000 samples on meshes with 100k to 2m vertices. The samples are computed with the constrained Poisson disk sampling scheme proposed in [CCS12].*

and we consider the polynomial

$$p_\rho(r) = \begin{cases} \frac{2}{\rho^3} r^3 - \frac{3}{\rho^2} r^2 + 1 & \text{for } r \leq \rho \\ 0 & \text{for } r > \rho \end{cases},$$

which is the unique cubic polynomial satisfying $p_\rho(0) = 1, \frac{\partial}{\partial r} p_\rho(0) = 0, p_\rho(\rho) = 0$, and $\frac{\partial}{\partial r} p_\rho(\rho) = 0$. Then, the matrix entries $\tilde{u}_{ij}$ of $\tilde{U}$ are given by

$$\tilde{u}_{ij} = p_\rho(d(v_i, v_{s_j})).$$

We choose the parameter ρ such that the support of every function contains a small number of sample points, e.g. $7 - 15$ sample points. The entries $\tilde{u}_{ij}$ can be interpreted as weights measuring the influence of sample v_{s_j} on vertex v_i of the mesh. Each sample only influences vertices in a local neighborhood. Within the neighborhood, the influence weights decrease as the geodesic distance

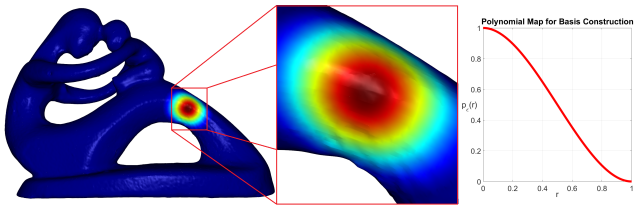

Figure 2: *Visualization of a locally supported basis function used for the construction of the subspace bases on the Fertility model. The image on the right shows a plot of the polynomial $p_\rho(r)$, see Equation (4), for $\rho = 1$.*

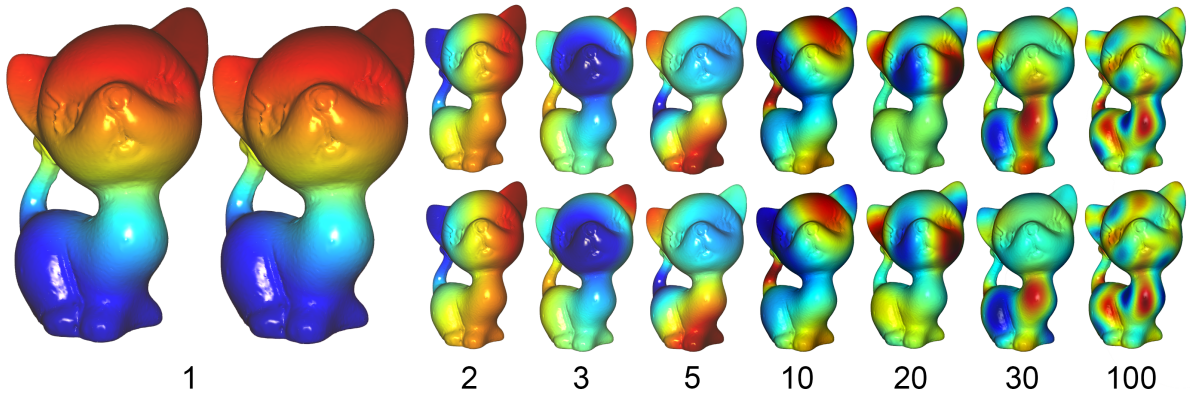

Figure 3: *Comparison of approximate (bottom row) and reference (top row) eigenfunctions of the Laplace–Beltrami operator on the Kitten model. The reference solutions are computed with MATLAB's sparse eigensolver.*

between the vertex and the sample increases. A visualization of a resulting function is shown in Figure 2.

We use a growing geodesic disk strategy for the construction of the locally supported functions. To obtain the j^{th} column of $\tilde{U}$, we start with the sample v_{s_j} and process the other vertices in order of increasing geodesic distance to v_{s_j}. For each visited vertex v_i, a triplet $< i, j, \tilde{u}_{ij} >$ is created. Once the distance to v_{s_j} is larger than ρ, all triplets for the j^{th} column are collected and the triplets for the next column are assembled. After the triplets for all columns are collected, the sparse matrix $\tilde{U}$ is generated.

The benefit of the growing disk strategy is that for the construction of the locally supported functions, only the local neighborhoods of the samples vertices are processed. In other words, this is a strategy for selecting exactly those pairs (v_i, v_{s_j}) that contribute non-zero entries to the sparse matrix $\tilde{U}$. As a consequence, the computational cost for the construction of $\tilde{U}$ depends on the number of non-zero entries. Since for a larger number on samples, we decrease the size of the support of the functions in a way that the number of entries of the matrix remains (approximately) constant, the computational cost is independent of the size of the space we construct. This can also be observed in Table 1, which lists timings for the individual steps of the basis construction.

We experimented with three different variants of the growing disk strategy that differ in the way they approximate the geodesic distance. The first variant is to use Dijkstra's single source algorithm, see [CLR90], on the edge graph of the mesh, where the weights of the edges are the edge lengths. An alternative is the use of the Short-Term Vector Dijkstra algorithm proposed in [CHK13]. This algorithm is a variant of Dijkstra's algorithms that corrects distance computations by unfolding the computed edge paths to a plane and measuring the distance in the plane. The third variant is to use Dijkstra's algorithm and correct the distances by taking the Euclidean distance in ambient $\mathbb{R}^3$ instead of the edge path distance. This variant is motivated by the fact that the diameter of the support of the functions we construct is quite small and for small distances, the Euclidean distance provides a good approximation of the geodesic distance. We refer to [SvWW00], where it is proven by Taylor expansion that squared Euclidean and the squared geodesic distances between two points agree up to third order in the geodesic distance of the points. We compared the three variants by looking at approximation quality of the resulting spectra as well as the required run time and found the third variant to provide the best trade-off.

In the last step of the basis construction, the matrix U is generated by normalizing the rows of $\tilde{U}$

$$u_{ij} = \frac{1}{\sum_{j=1}^{d} \tilde{u}_{ij}} \tilde{u}_{ij}.$$

This step ensures that the functions U form a partition of unity on the surface. Though, we did not encounter such a situation in our experiments, it is possible that a vertex of the mesh is not in the support of any of the functions. This case could be dealt with by adding a functions centered the vertex to the basis.

Restricted eigenproblem The subspaces are designed such that when the subspace dimension d is large enough, low- and mid-frequency functions can be well-approximated. To get approximations of the lowest m eigenvalues and eigenfunctions, we restrict the optimization problem (5) to the subspace spanned by U. This means that instead of searching for a minimizer in the set of all matrices $\Phi \in \mathbb{R}^{n \times m}$, we restrict the search space to matrices $\bar{\Phi} \in \mathbb{R}^{n \times m}$ that have the form $\bar{\Phi} = U\bar{\phi}$, where $\bar{\phi}$ is a matrix in $\mathbb{R}^{d \times m}$. Our experiments, we chose $d = 2m$.

To formulate the restricted optimization problem, we use the restricted mass and stiffness matrices

$$\bar{M} = U^T M U \qquad \text{and} \qquad \bar{S} = U^T S U.$$

Then the restriction of (5) is

$$\min_{\bar{\phi} \in \mathbb{R}^{d \times m}} \operatorname{tr}\left(\bar{\phi}^T \bar{S} \bar{\phi}\right) \tag{7}$$
$$\text{subject to } \bar{\phi}^T \bar{M} \bar{\phi} = Id.$$

The columns $\bar{\phi}_i$ of the resulting minimizer $\bar{\phi}$ are the restricted eigenvectors. The corresponding restricted eigenvalues are $\bar{\lambda}_i =$

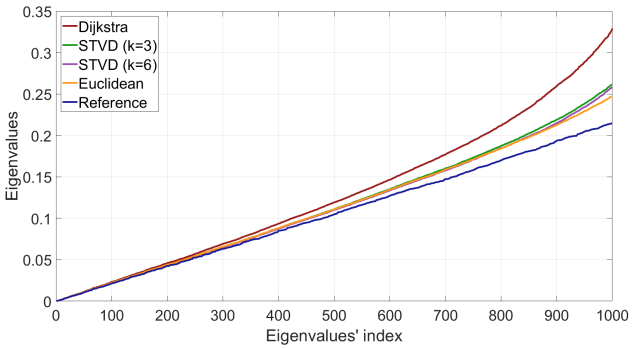

Figure 4: *Approximations of the lowest 500 eigenvalues for different subspace constructions on the Fertility model: The subspace constructions differ in the scheme that is used for the approximation of the geodesic distances. The schemes used are: Dijkstra's algorithm, Dijkstra's algorithm with Euclidean distance correction, and Short Term Vector Dijkstra (STVD) with two different parameter settings. The subspace used is 1000-dimensional. All 1000 eigenvalues are shown, though, we recommend using only the first 500.*

$\bar{\phi}_i^T \bar{S} \bar{\phi}_i$. The pairs of $(\bar{\lambda}_i, \bar{\phi}_i)$ satisfy the equation

$$\bar{S}\bar{\phi}_i = \bar{\lambda}_i \bar{M} \bar{\phi}_i. \tag{8}$$

The eigenvectors $\bar{\phi}_i$ are d-dimensional vectors listing coordinates with respect to the basis U. The vectors can be lifted to nodal vectors

$$\bar{\Phi}_i = U \bar{\phi}_i \tag{9}$$

that describe functions on the mesh, which we call the *restricted eigenfunctions*.

Solving the eigenproblem The restricted problem is a low-dimensional eigenproblem involving sparse matrices. Explicitly, the matrices $\bar{M}$ and $\bar{S}$ have 29 and 32 non-zero entries per row for the Kitten model in a 1000-dimensional subspace. We observed similar numbers for other models and subspace dimensions. We experimented with solvers for sparse matrices and GPU-based dense solvers for the restricted problem, developing a preference for the latter. The dense solver computes all d eigenvectors of the reduced problem in a reasonable time for sizes up to $d = 10k$. Timings are listed in Table 1.

Storage requirements In addition to the accelerated basis construction, less data is required to represent the approximate bases. For spectral methods, this implies less storage is required. Hence larger bases can be used in-core and evaluations, like reconstructing a shape from the reduced representation, are faster. Explicit timings for the latter can be found in the paragraph on simulation in modal coordinates in Section 5. Since the eigenfunctions have dense nodal vectors, for a mesh with n vertices, storing m (unreduced) eigenvectors requires $\mathcal{O}(nm)$ storage. The approximate eigenvectors are represented by two matrices: the sparse matrix U representing the subspace basis and a dense matrix representing the reduced coordinates of the eigenvectors. Since we choose the size of the support of the subspace basis functions such that in average about 10 samples influence each vertex, the matrix U has on average 10 entries per row. Hence, U requires $\mathcal{O}(n)$ storage, which is independent of d, the dimension of the space. The intuition is that if more samples are used, the support of the basis functions shrinks. Since we choose d to be $2m$, the matrix storing the reduced coordinates of the approximate eigenfunctions requires $\mathcal{O}(m^2)$ storage. Together, the two matrices require $\mathcal{O}(n+m^2)$ storage. For example, representing $5k$ eigenfunctions on a mesh with $1M$ vertices in double (8 byte) precision requires $5k * 1M * 8$ byte $= 40$ GB storage. An approximate basis computed in a 10k-dimensional subspace requires about $(10 * 1M + 10k * 5k) * 8$ byte < 0.5 GB storage.

Properties of the restricted eigenfunctions In the following, we discuss properties of the restricted eigenfunctions. First, we show that the lifted restricted eigenfunctions, given by nodal vectors $\bar{\Phi}_j$, are L^2-orthonormal on the full resolution mesh.

Lemma 1 The restricted eigenfunctions are pairwise L^2-orthonormal.

Proof Using the definition of the restricted eigenfunctions (9), we obtain

$$\left\langle \bar{\Phi}_i, \bar{\Phi}_j \right\rangle_{L^2} = \bar{\Phi}_i^T M \bar{\Phi}_j = \bar{\phi}_i^T U^T M U \bar{\phi}_j = \bar{\phi}_i^T \bar{M} \bar{\phi}_j = \delta_{ij},$$

where δ_{ij}, the Kronecker delta, is 0 for $i \neq j$ and 1 for $i = j$. For the last step, we use the property that the solutions (8) of the minimization problem (7) are orthonormal with respect to $\bar{M}$. □

The second property we want to discuss relates to the Dirichlet energy of the restricted eigenfunctions. The Dirichlet energy of a functions f is the quadratic functional $\langle f, f \rangle_{H_0^1}$ associated to the bilinear form (2). Equation (3) implies that, for (L^2-normalized) eigenfunctions of the Laplace–Beltrami operator, the eigenvalue agrees with the Dirichlet energy of the corresponding eigenfunction. We show that an analogous relation holds true for the restricted eigenfunctions and eigenvalues.

Lemma 2 The Dirichlet energy of the i^{th} restricted eigenfunction equals the restricted eigenvalue $\bar{\lambda}_i$.

Proof Using the definition of the restricted eigenfunctions (9), we get

$$\left\langle \bar{\Phi}_i, \bar{\Phi}_i \right\rangle_{H_0^1} = \bar{\Phi}_i^T S \bar{\Phi}_i = \bar{\phi}_i^T U^T S U \bar{\phi}_i = \bar{\phi}_i^T \bar{S} \bar{\phi}_i = \bar{\lambda}_i \bar{\phi}_i^T \bar{M} \bar{\phi}_i = \bar{\lambda}_i.$$

In the last step, we used the orthogonality constraint of the restricted eigenvalue problem (7). □

A consequence of Lemma 2 is that if the restricted eigenvalues $\bar{\lambda}_i$ are close to the eigenvalues λ_i, then the Dirichlet energies of the corresponding restricted and unrestricted eigenfunctions, $\bar{\Phi}_i$ and Φ_i, are close.

The restricted eigenfunctions $\bar{\Phi}_i$ can be written as a linear combination of the eigenfunctions Φ_k

$$\bar{\Phi}_i = \sum_k a_{ik} \Phi_k, \tag{10}$$

with Fourier coefficients $a_{ik} = \left\langle \bar{\Phi}_i, \Phi_k \right\rangle_{L^2}$. Since the $\bar{\Phi}_i$ have unit L^2-norm, the coefficients satisfy

$$\sum_k a_{ik}^2 = 1 \tag{11}$$

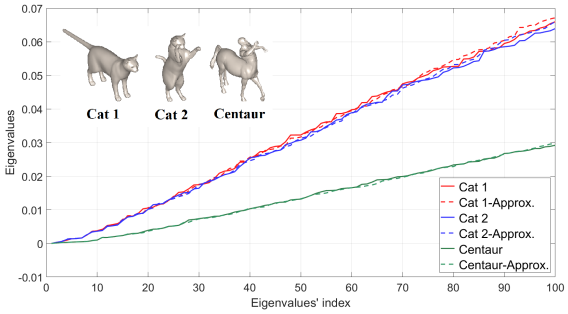

Figure 5: *Comparison of the first 100 eigenvalues, approximation and reference solution, for three models, two Cat and one Centaur models. Reference solutions are computed using MATLAB's sparse eigensolver.*

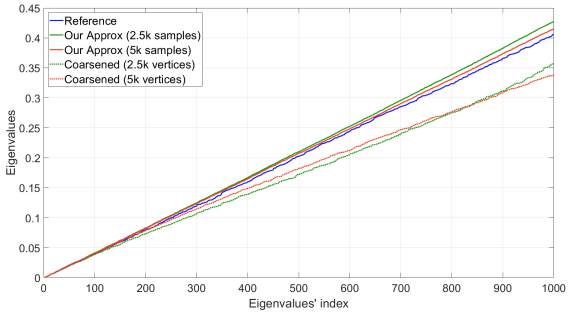

Figure 6: *Approximations of the first 1000 eigenvalues of the Laplace–Beltrami operator on the Chinese Dragon mesh. The reference solution (blue), computed with MATLAB's sparse eigensolver, is compared with approximations computed with the proposed scheme with 2k (green) and 5k (red) dimensional subspaces and also computed from coarsened meshes with 2k (dashed green) and 5k (dashed red) vertices.*

for any i. Using Lemma 2, we get a relation of the Fourier coefficients a_{ik} of the restricted eigenfunctions, the restricted eigenvalues $\bar{\lambda}_i$ and the unrestricted eigenvalues λ_k.

Theorem 3 The Fourier coefficients of the restricted eigenfunctions satisfy

$$\bar{\lambda}_i = \sum_k a_{ik}^2 \lambda_k. \tag{12}$$

Proof Using Lemmas 1 and 2, we get

$$\bar{\lambda}_i = \bar{\Phi}_i^T S \bar{\Phi}_i = (\sum_l a_{il}\Phi_l)^T S (\sum_k a_{ik}\Phi_k) = (\sum_l a_{il}\Phi_l)^T \sum_k a_{ik}\lambda_k M\Phi_k$$
$$= \sum_l \sum_k \lambda_k a_{il} a_{ik} \Phi_l{}^T M \Phi_k = \sum_k \lambda_k a_{ik}^2,$$

which proves the theorem. □

The combination of equations (11) and (12) indicates that any restricted eigenfunction $\bar{\Phi}_i$ is a linear combination of eigenfunctions Φ_k with eigenvalues λ_k close to the restricted eigenvalue $\bar{\lambda}_i$.

5. Experiments

We implemented our approximation algorithm using the Eigen [GJ*10] and LibIGL [JP*16] libraries. For solving the restricted, low-dimensional eigenproblems, we use the GPU-based solver provided by the cuSOLVER library.

Approximation In our experiments, we found that the spectrum computed with the proposed scheme approximates well the spectrum of a numerical reference solution, which is computed with MATLAB's sparse eigensolver. Figure 5 shows plots of the first one hundred reference and approximate eigenvalues for three different models. Figure 4 shows plots of the first 1000 eigenvalues of the reference solutions as well as approximations that are computed in a 1000-dimensional subspace using different schemes for the bases constructions. Note that while the figure shows all 1000 eigenvalues, we recommend using only the first half of the computed eigenvalues, which in this case are the first 500 eigenvalues. The figure stills shows all eigenvalues to give a better comparison. The schemes for basis construction that are compared differ in the way the geodesic distance is approximated. Results using Dijkstra's graph distances, the Dijkstra distances with Euclidean distance as correction and the Short Term Vector Dijkstra with two different parameter settings are shown. The figure illustrates that all four variants produce good approximation results. Taking the timings, shown in Table 3, into account, we favored the Euclidean correction of Dijkstra's algorithm for our experiments.

To evaluate the approximation of the eigenvectors, we compute the Fourier coefficients of the approximate eigenfunctions $\bar{\Phi}_i$ with respect to the reference eigenbasis $\{\Phi_k\}$, see Equation (10). Figure 7 and a supplementary video show plots of Fourier coefficients for an approximate basis computed in the 1000-dimensional space on the Kitten model. While for the lower eigenfunctions we observe a sharp peak at the index of the eigenfunction, the higher approximate eigenfunctions are a linear combination of reference eigenfunctions with similar eigenvalue. To put this result into a broader context, we want to point the reader to the supplementary

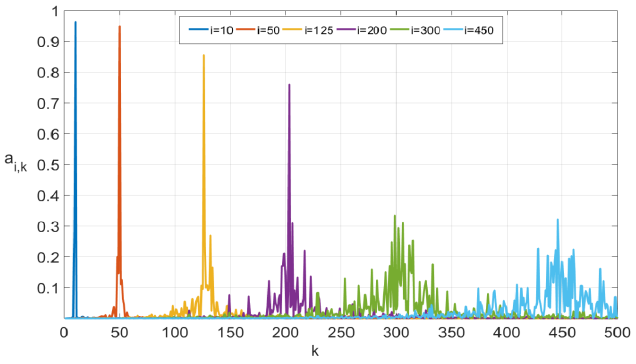

Figure 7: *Plots of the Fourier coefficients, $a_{ik} = \langle \bar{\Phi}_i, \Phi_k \rangle_{L^2}$, of restricted eigenfunctions $\bar{\Phi}_i$ in the reference eigenbasis $\{\Phi_k\}$. The $\bar{\Phi}_i$ are computed in a 1000-dimensional space on the Kitten model.*

Model	Vertices	Subsp. dim.	Eigen-pairs	M,S	Sam-pling	Adja-cency	Basis	$\bar{M},\bar{S}$	Solve eigenp.	Total	Reference
Chinese-Dragon	127K	1K	500	0.42	0.11	0.19	0.27	0.23	1.39	2.61	225.62
Kitten	137K	1K	500	0.42	0.12	0.15	0.24	0.17	1.34	2.44	242.99
Fertility	241K	1K	500	0.61	0.24	0.25	0.44	0.34	1.37	3.26	452.34
Red Circular Box	701K	1K	500	1.60	1.69	0.85	1.18	0.95	1.39	7.67	1196.37
Isidore Horse	1104K	1K	500	2.30	3.23	1.28	1.92	1.58	1.41	11.63	2001.12
Neptune	2003K	1K	500	4.34	4.65	2.79	4.32	3.30	1.31	20.70	Memory bound
Chinese-Dragon	127K	5K	2.5K	0.42	0.13	0.19	0.47	0.23	52.18	53.62	4000.01
Kitten	137K	5K	2.5K	0.42	0.12	0.15	0.49	0.19	55.94	57.31	5932.93
Fertility	241K	5K	2.5K	0.62	0.19	0.25	1.02	0.32	56.58	58.99	10,707.84
Red Circular Box	701K	5K	2.5K	1.52	0.61	0.83	2.92	1.08	50.20	57.17	Memory bound
Isidore Horse	1104K	5K	2.5K	2.39	1.15	1.29	4.72	1.49	52.54	63.60	Memory bound
Neptune	2003K	5K	2.5K	4.60	4.76	2.82	9.06	3.58	56.21	81.03	Memory bound
Chinese-Dragon	127K	10K	5K	0.43	0.17	0.20	0.80	0.25	421.86	423.71	15,479.83
Neptune	2003K	10K	5K	4.53	2.88	2.80	15.58	3.47	434.36	463.64	Memory bound

Table 1: *Timings (in seconds) for the individuals steps of the proposed approximation algorithm. From left to right: number of vertices of the mesh, dimension of subspace, number of eigenpairs computed, construction of matrices M and S, sampling stage, construction of vertex neighborhoods, construction of subspace basis functions, computation of reduced matrices $\bar{M}$ and $\bar{S}$, solving restricted eigenproblem, total time for approximation algorithm, and comparison timings of MATLAB's eigensolver for the same setting.*

material that includes an experiment in which we explore how the Laplace–Beltrami eigenfunctions change when the metric of the kitten model is slightly altered. In a second experiment, we evaluate how well the space spanned by the approximate eigenfunctions can approximate the reference eigenfunctions. Figure 8 shows a plot of the norms of the difference between reference eigenfunction Φ_k and its projection onto the space spanned by the first 500 approximate eigenfunctions $\bar{\Phi}_i$. The figure additionally shows an analogous plot where the roles of the approximate and the reference eigenfunctions are exchanged. For a visual comparison, Figure 3 shows color plots of some of the reference and approximate eigenfunctions on the kitten mesh.

Comparison to mesh coarsening Mesh coarsening can be used for the fast approximation of the eigenvalues of the Laplace–Beltrami operator. Instead of computing the spectrum on the full-resolution mesh, the mesh is coarsened and the spectrum of the coarse mesh is computed. Figure 6 shows 1000 reference eigenvalues (computed using MATLAB's sparse eigensolver on the full-resolution mesh), approximations obtained with our scheme using 2k and 5k dimensional subspaces, and eigenvalues computed from simplified meshes with 2k and 5k vertices. The figure demonstrates the benefits in approximation quality of our scheme compared to the mesh simplification scheme. The computation times for both, our approximation scheme and mesh coarsening, are comparable as the most expensive step for both schemes is solving the low-dimensional eigenvalue problem.

An essential difference between mesh coarsening and our approach is that our scheme solves an eigenproblem in a subspace of the function space on the full-resolution mesh. Therefore, we can use restrictions on the matrices M and S. In contrast, the mesh coarsening scheme creates a new function space and new matrices. As a consequence, our scheme results in approximate eigenfunctions on the full-resolution mesh. By construction the functions are L^2-orthonormal on the full-resolution mesh and their Dirichlet energy agrees with the approximate eigenvalues. We refer to Section 4

Solver	MATLAB (500)	MATLAB (1000)	CUDA (1000)
Time	5.30	21.45	1.40

Table 2: *Performance of MATLAB's sparse eigensolver vs. the GPU-based dense solver from the cuSOLVER library for computing the first 500 (left) and all 1000 (middle and right) eigenvalues of the restricted 1000-dimensional eigenvalue problem.*

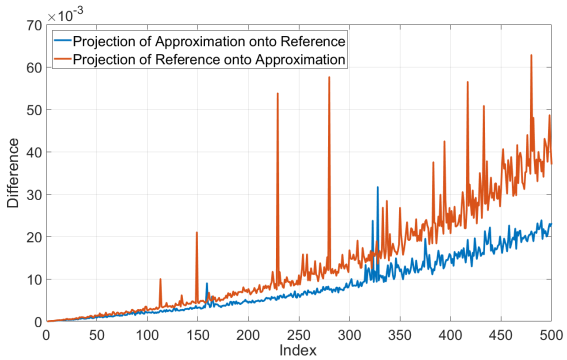

Figure 8: *The norms of the difference between reference eigenfunction Φ_k and its projection onto the space spanned by the first 500 approximate eigenfunctions $\bar{\Phi}_i$ for $k \in \{1,2,...,500\}$ are shown in red and an analogous plot with roles of Φ_k ad $\bar{\Phi}_i$ exchanged in blue.*

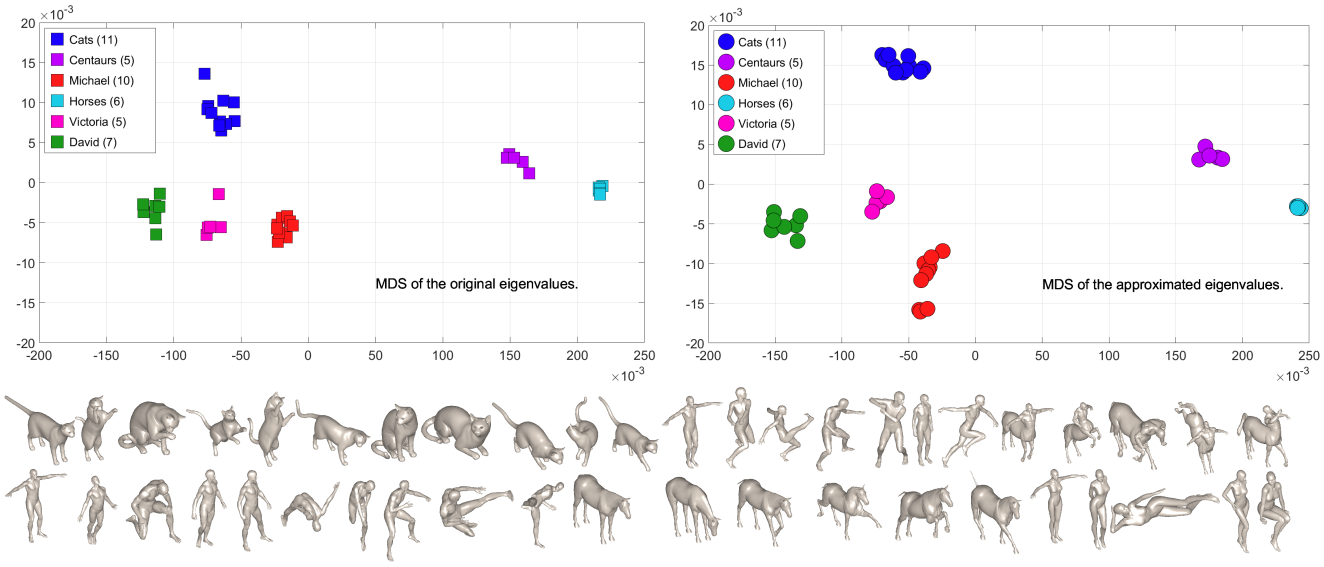

Figure 9: *Plots visualizing the shape distances of a collections of models. The shape distances are computed from the Shape DNA, approximated with the proposed method (top-right) and reference eigenvalues (top-left) computed using MATLAB's sparse eigensolver. The plots are generated with multi-dimensional scaling applied to the matrix containing all pairwise distances. Visualization of the models, which are taken from the TOSCA data set, are shown at the bottom.*

for a discussion of the properties. In contrast, the mesh coarsening scheme computes eigenfunction in a function space on the coarse mesh. These eigenfunctions could be mapped to functions on the full-resolution mesh, but the resulting functions would not be L^2-orthonormal and lose their connection to the approximate eigenvalues.

Chuang et al. [CLB*09] introduced a scheme for estimating the Laplace–Beltrami operator that constructs a coarse voxel grid containing the surface mesh and creates a function space by restricting a set 2nd-order tensor-product B-splines defined on the voxel grid to the surface mesh. The dimension of the function space depends on the voxel grid that is used and is independent of the resolution of the surface mesh. This approach can be used to approximate the eigenvalues of the Laplace–Beltrami operator. To put this approach in context to our scheme, we want to note that similar to the mesh coarsening approach, the scheme constructs a new function space and new mass and stiffness matrices. In particular, the approximate eigenfunctions are elements of the function space induced by the voxel grid. This means, to use them for spectral methods, the functions need to be mapped to the function space of the mesh and the resulting functions will not be L^2-orthonormal anymore.

Computation times Computation times for the individual steps of the proposed approximation algorithm for different numbers of eigenpairs and sizes of meshes are shown in Table 1. For reference, timings of MATLAB's ARPACK-based large-scale sparse eigensolver for the same configurations are shown. For the some problems the MATLAB solver failed because it reached the bound of the available main memory. This could be avoided by using an out-of-core implementation, as discussed in [VL08]. However, this comes at the cost of much higher computation times. The approximation algorithm, on the other hand, requires less memory, since only reduced coordinates need to be stored for every eigenvector, and we can solve all the problems listed in the table in-core.

For the *Fertility* model with 241k vertices, the computation of 500 and 5000 eigenfunctions takes 3.3 and 59 seconds. For the computation of 500 restricted eigenfunctions all steps require a substantial part to the total time. However, for 2500 eigenfunctions and more, solving the restricted eigenproblem is the most expensive step.

We experimented with different solvers for the restricted eigenproblem. Since the restricted stiffness and mass matrices are still sparse matrices, we experimented with sparse and dense solvers for the low-dimensional eigenvalue problems. Table 2 shows representative times we obtained with MATLAB's sparse eigensolver and a GPU-based dense solver from the cuSOLVER library. Based on the run times obtained in our experiments, we recommend using the dense solver.

	Dijkstra	Euclidean	STVD (k=3)	STVD (k=6)
Basis Construction	2.72	2.73	7.07	14.66

Table 3: *Timing of basis construction with different methods for the approximation of the geodesic distance on the mesh. From left to right: Dijkstra's algorithm, Dijkstra's algorithm with Euclidean distance correction, Short term vector Dijkstra with window size 3, Short term vector Dijkstra with window size 6.*

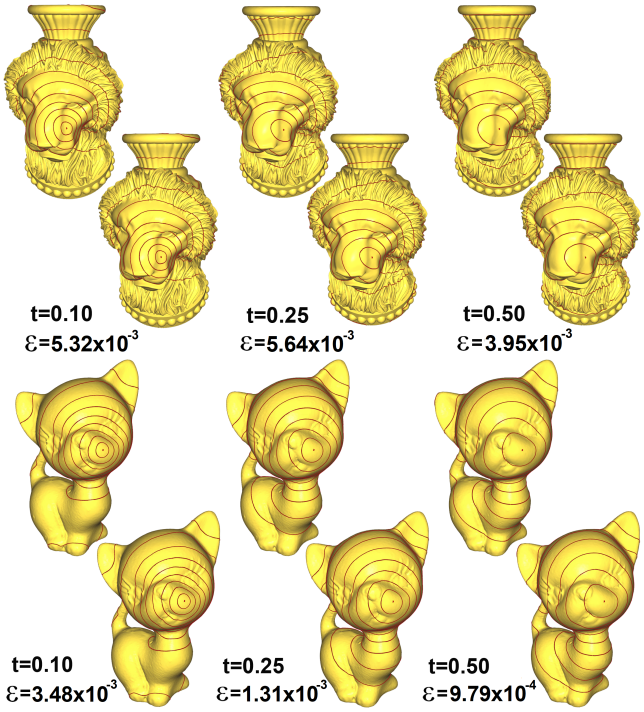

Figure 10: *Comparison of diffusion distances computed using reference eigenpairs (first and third rows) and our approximations (second and fourth rows) on the Vase-Lion and Kitten models. Parameter of the diffusion distance, t, and relative L^2 approximation error, ε, are shown.*

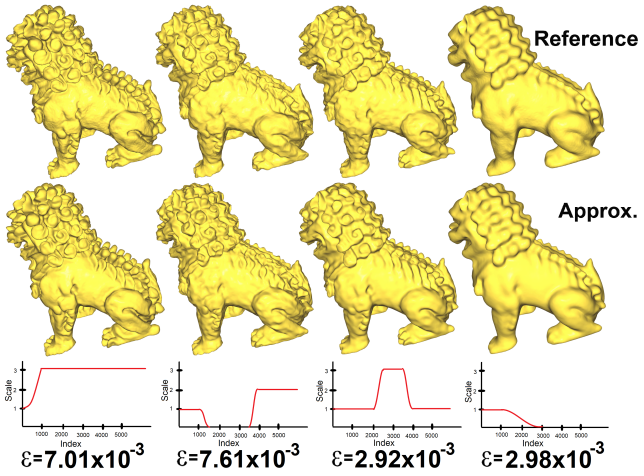

Figure 11: *Comparison of results of spectral mesh filtering on the Chinese-Dragon model using 5000 approximated eigenvalues and eigenfunctions (bottom row) and 5000 eigenvalues and eigenfunctions computed using MATLAB (top row). Relative L^2 approximation errors, ε, are shown.*

6. Applications

To evaluate the proposed approximation scheme, we use the approximate eigenvalues and eigenfunctions for different spectral methods and compare the results to those obtained with reference eigenvalues and eigenfunctions, which we compute with MATLAB's eigensolver.

Shape DNA Shape DNA [RWP06] is a simple, yet effective, spectral shape descriptor. The simplicity makes it well-suited for our comparison. Figure 9 compares results obtained with the approximate and the reference spectra, where the lowest 100 eigenvalues are used for the Shape DNA. The test dataset, which is part of the TOSCA data set [BBK08], consists of a variety of shapes with different poses for each of the shapes. The Shape DNA of each shape and the pairwise shape distances are computed. The resulting distances are visualized by an MDS projection to the plane. The figure illustrates that the approximate eigenvalues yield comparable results to the reference solution. For three examples shapes, hundred approximate and reference eigenvalues are shown in Figure 5. The figure illustrates that the differences between approximate and reference spectra are small compared to the difference of eigenvalues of different shapes.

Diffusion distance The second spectral method we consider is the diffusion distance [NLCK05]. We chose this spectral distance because the computation combines eigenvalues and eigenfunctions and the distance has fewer parameter than alternatives, such as the heat kernel signature. The latter makes it easier to compare results. Figure 10 shows results for two models and different values of the parameter t. The results obtained using the lowest 1000 approximate eigenvalues and eigenfunctions and reference eigenvalues and eigenfunctions are shown. Furthermore, the relative L^2 approximation error, denoted by ε, of the differences between approximate and reference result is shown. The visual comparison indicates that the resulting distances are quite close, which is confirmed by the computed approximation errors. As discussed in Section 5, the precomputation time for the approximate solution is two orders of magnitude shorter.

Mesh filtering Figures 11 and 12 show the results of spectral filtering [VL08] using the approximate eigenvalues and eigenfunctions. For comparison, results obtained using the reference eigenvalues and eigenfunctions are shown. The visual comparison shows that results of comparable quality can be obtained using the approximate basis. The benefit of using the approximate basis is the reduced precomputation time. This is a crucial factor as with the approximation the overall time needed for spectral filtering becomes comparable to the time required by alternative state-of-the-art mesh filtering schemes. Moreover, with the approximation, the filtering approach only requires solving a low-dimensional dense eigenproblem, which is in contrast to many recent filtering schemes that are based on large scale, non-convex optimization problems. The benefits that spectral filters can enhance frequencies and can be interactively modified are preserved. Due to the lower storage requirements, the approximation method can also process larger

meshes and more eigenfunctions in-core than the original scheme using unreduced eigenfunctions. Though approximation of the reference solution is not a quality criteria for the result of filtering, we are listing the approximation errors since they evaluate how-well the low frequency subspaces are approximated.

Parameter-dependent operators Recent work [HSvTP10, CSBK16, LJC17, WBPS17] indicates that spectral methods can profit from using not only the Laplacian but also other operators. For example, the Laplacian can be augmented with a potential that includes additional information about the extrinsic curvature. The operators considered usually depend on one or more parameters. For example, a parameter that weights the importance of extrinsic information against the importance of intrinsic information. The proposed approximation algorithm provides the possibility to efficiently explore such parameter spaces. Once the subspace basis is constructed and the reduced matrices of the relevant operators are generated, the approximation of the lowest 100 eigenpairs for different parameter settings can be computed at interactive rates. For example, a user can change the parameters and receive interactive feedback, such as visualizations of the eigenfunctions at the current parameter setting. Figure 13 shows examples of results obtained with the family of operators described in [Sch13, pages 72–73], which extends the construction of a modified Laplacian from [HSvTP10]. In the discrete setting, the matrix A with entries

$$A_{ij} = \langle N(v_i), N(v_j) \rangle S_{ij}$$

is contructed, where S_{ij} are the entries of the cotangent matrix and $N(v_i)$ is the normal at vertex v_i. Then a one-parameter family of operators is defined as

$$(1-t)S + tA. \tag{13}$$

The operators are intrinsic for $t = 0$ and a potential, dependent on the extrinsic curvature, is blended in for $t > 0$. Figure 13 shows examples of approximate eigenfunctions for different values of t.

Simulation of elastic deformables In the following, we will discuss how our approach can be used for fast approximation of *vibration modes* of deformable objects. Vibration modes are widely used in graphics for fast simulation [BJ05, vTSSH13], simulation-based shape editing [HSvTP11], sound synthesis [CAJ09], editing simulations [BSG12], and for deformable motion design [HSvTP12a]. Yang et al. [YLX*15] propose a model reduction for the simulation of deformable objects that involves the construction of a linear subspace. They state that solving the eigenproblem to obtain vibration modes is computationally expensive and they therefore use a Krylov subspace method to compute linear inertia modes instead. Our algorithm would enable the method proposed in [YLX*15] to compute approximate vibration modes in less time than what is needed for their construction of the linear inertia modes.

We want to emphasize that the approximations are solutions of a restricted eigenvalue problem. Hence the eigenvalues are physically meaningful and the approximate eigenvalues and eigenmodes can be used for simulation in modal coordinates. Moreover, the reduced storage requirements, which were discussed in Section 4, also extend to the approximate vibration modes.

We consider a discrete elastic object modeled by a triangle or tetrahedral mesh and describe the configurations of the object by $3n$-dimensional vectors listing the coordinates of all n vertices. We look at a rest configuration x_0 of the object and use displacement vectors u to describe deformed configurations. The energy stored in any configuration $x = x_0 + u$ is measured by an elastic potential $E(x)$. The vibration modes are the eigenfunctions Φ_i to the generalized eigenvalue problem

$$H\Phi_i = \lambda M \Phi_i, \tag{14}$$

where $H = \frac{\partial^2}{\partial x^2} E(x_0)$ is the Hessian of the elastic energy at x_0 and M is the mass matrix. The linearized equations of motion of the discrete elastic object are

$$M\ddot{u}(t) + (\alpha M + \beta H)\dot{u}(t) + Hu(t) = F, \tag{15}$$

subject to suitable initial conditions on positions $u(0)$ and velocities $\dot{u}(0)$. Here F represents external forces. We can express any displacement u as a superposition of eigenmodes, $u = \Phi q$, where Φ

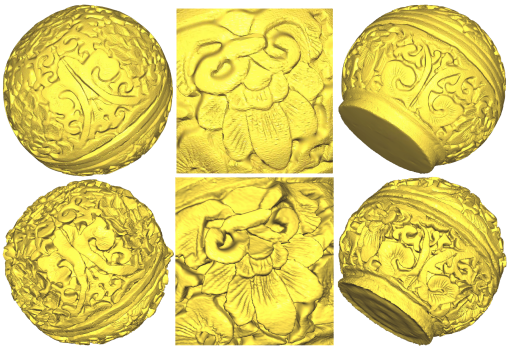

Figure 12: *Results of a sharpening filter using 5000 approximate eigenpairs on the Red Circular Box model. Top row shows input and bottom row results.*

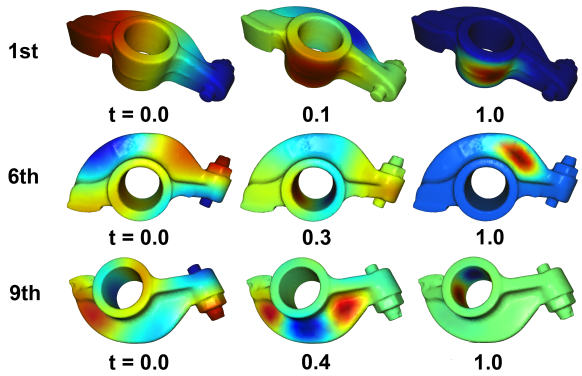

Figure 13: *Eigenfunctions of a one-parameter family of operators are shown. For $t = 0$, the operator is the Laplace–Beltrami operator. For other values of t the operator includes extrinsic information. The higher the value of t the stronger the influence of the extrinsic information. The proposed approximation scheme allows to interactively explore the eigenfunctions the one-parameter family of operators.*

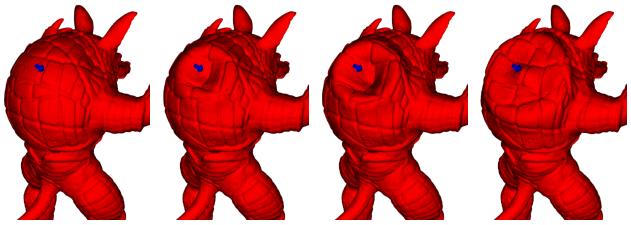

Figure 14: *Real-time simulation of an elastic deformable (165k vertices) using 500 approximated vibration modes. A strong point force is applied to the back. High frequent details in the dynamics that spread across the mesh without (numerically) dissipating are observed.*

is the matrix whose columns are the vibration modes Φ_i and q is the vector listing the modal coordinates of u. In the basis of vibration modes, the equations of motion decouple to $3n$ independent ODEs

$$\ddot{q}_i(t) + (\alpha + \beta\lambda_i)\dot{q}_i(t) + \lambda_i q_i(t) = \left(\Phi^T F\right)_i \tag{16}$$

to which analytic solutions are known. Being able to compute analytic solutions has many benefits over numerical integration. For example, the solution at any point in time can be computed without time-step restrictions and there is no numerical damping. For efficiency, not all vibration modes but only the lowest m are computed and used for the simulation. Then, Φ is the $3n \times m$ matrix storing the first m vibration modes as its columns.

Our approach offers different benefits for the simulation in modal coordinates. One is that the computational cost for constructing the modal basis is significantly reduced. Moreover, the transformation from modal coordinates q to world coordinates u is faster. The reason is that the matrix $\bar{\Phi}$ storing the first m approximate vibration modes can be decomposed $\bar{\Phi} = U\bar{\phi}$, where $U \in \mathbb{R}^{3n\times d}$ is the sparse matrix storing the subspace basis and $\bar{\phi} \in \mathbb{R}^{d\times m}$ is the matrix representing the approximate vibration modes in reduced coordinates. This means that instead of a matrix-vector product with a dense $3n \times m$-matrix, only products with a dense $d \times m$-matrix and a sparse $3n \times d$-matrix are needed to transform from modal to world coordinates. The third benefit is that the approximate modal basis requires less memory since only the low-dimensional dense matrix $\bar{\phi}$ and the sparse matrix U are stored instead of the matrix Φ. This is crucial when memory requirements for storing Φ exceed the GPU storage space.

In a supplementary video we show an evaluation of this application, where we simulate the Armadillo model (unsimplified, 165k vertices) using 500 approximated vibration modes, which were computed using our subspace construction from $d = 1000$ samples. In Figure 14 we show snapshots of this simulation. To highlight the fine resolution of the dynamics that can be expressed using this approach, we set the damping quotients α and β to very low values and "poke" the mesh by shortly applying large external forces to a single vertex (shown by a blue arrow). The resulting simulation shows the advantages of using vibration modes to reduce and solve the linearized equations of motion, as we observe high frequent details in the dynamics as well as no dissipation due to numerical damping, such that even small shock waves can slowly propagate across the entire mesh. The computation of the approximated vibration modes is only 15.2 seconds, whereas computing 500 full vibration modes using *MATLAB* takes 30.1 minutes on the same machine. For a quantitative comparison of the simulation using fully computed vibration modes to the simulation using our approximated vibration modes, we computed the relative error between the two simulations, that is $e^i_{\text{rel}} := \|x(i\cdot\delta) - \tilde{x}(i\cdot\delta)\|_M / \|x(i\cdot\delta)\|_M$. Here, δ is the time-step, i the index of the frame, and $x(t)$ and $\tilde{x}(t)$ are the vertex positions of the solutions to (16) using fully computed vibration modes and approximated vibration modes respectively. For the first 250 frames of the simulation shown in Figure 14, we got an average relative error of 0.00698 and a maximal relative error of 0.01141. Visually, the two simulations are indistinguishable. Updating the current state using the product $U \cdot \left(\bar{\phi} \cdot q(t)\right)$ requires around 7 milliseconds, whereas the product $\Phi q(t)$ takes 120 milliseconds of computation time on average, which prohibits real-time applications.

7. Conclusions

We present a fast approximation algorithm for the lowest part of the spectrum of the Laplace–Beltrami operator. Our experiments demonstrate that the approximate spectra are close to the reference spectra, which were computed with state-of-the-art large-scale sparse eingensolvers. We also show that spectral methods produce comparable results with the approximate and the reference spectra. The benefit of the approximation algorithm is the lower computational cost, which reduces precomputation time for spectral methods by two order of magnitude and thereby make spectral methods even more attractive for geometry processing applications. A second benefit is the computation of the approximate spectra does not require a sophisticated large-scale sparse eigensolver, but only requires to solve a low- dimensional eigenproblem.

Concerning future work, we see potential that the proposed approach can be extended to a fast approximation algorithm for the computation of compressed manifold modes [NVT*14] and compressed vibration modes [BH17]. Due the sparsity enforcing term that is integrated to the eigenproblem, the computation of compressed modes poses a challenging problem. Moreover, we think that reduced simulation in modal coordinates can benefit from the proposed approach. Since computation times for approximating vibration modes are greatly reduced, one direction would be to use the approach for basis update in reduced non-linear elastic simulation. Furthermore, the fact that the approximate modes can be efficiently stored and processed makes the method interesting for sound synthesis as more eigenfunctions produce richer sound.

Acknowledgements

We thank Timothy Balint for proof-reading a draft of this paper. This project is supported in part by the Indonesian Endowment Fund for Education (LPDP) via a doctoral scholarship for Ahmad Nasikun.

References

[ABCCO13] AZENCOT O., BEN-CHEN M., CHAZAL F., OVSJANIKOV M.: An operator approach to tangent vector field processing. *Computer Graphics Forum 32*, 5 (2013), 73–82. 2

[AOCBC15] AZENCOT O., OVSJANIKOV M., CHAZAL F., BEN-CHEN M.: Discrete derivatives of vector fields on surfaces – an operator approach. *ACM Trans. Graph. 34*, 3 (2015), 29:1–29:13. 2

[ASC11] AUBRY M., SCHLICKEWEI U., CREMERS D.: The wave kernel signature: A quantum mechanical approach to shape analysis. In *ICCV* (2011), pp. 1626–1633. 2

[AW11] ALEXA M., WARDETZKY M.: Discrete Laplacians on general polygonal meshes. *ACM Trans. Graph. 30*, 4 (2011), 102:1–102:10. 3

[Bat13] BATHE K.-J.: The subspace iteration method - revisited. *Computers and Structures 126* (2013), 177–183. 3

[BBGO11] BRONSTEIN A. M., BRONSTEIN M. M., GUIBAS L. J., OVSJANIKOV M.: Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Trans. Graph. 30*, 1 (2011), 1:1–1:20. 2

[BBK08] BRONSTEIN A. M., BRONSTEIN M. M., KIMMEL R.: *Numerical geometry of non-rigid shapes*. Springer, 2008. 10

[BBL*17] BRONSTEIN M. M., BRUNA J., LECUN Y., SZLAM A., VANDERGHEYNST P.: Geometric deep learning: Going beyond euclidean data. *IEEE Signal Process. Mag. 34*, 4 (2017), 18–42. 2

[BH17] BRANDT C., HILDEBRANDT K.: Compressed vibration modes of deformable bodies. *Computer Aided Geometric Design 52–53* (2017), 297–312. 12

[BJ05] BARBIČ J., JAMES D. L.: Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Trans. Graph. 24*, 3 (2005), 982–990. 11

[BMM*15] BOSCAINI D., MASCI J., MELZI S., BRONSTEIN M. M., CASTELLANI U., VANDERGHEYNST P.: Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. *Computer Graphics Forum 34*, 5 (2015), 13–23. 2

[BSEH17] BRANDT C., SCANDOLO L., EISEMANN E., HILDEBRANDT K.: Spectral processing of tangential vector fields. *Computer Graphics Forum 36*, 6 (2017), 338–353. 2

[BSEH18] BRANDT C., SCANDOLO L., EISEMANN E., HILDEBRANDT K.: Modeling *n*-symmetry vector fields using higher-order energies. *ACM Trans. on Graph. 37*, 2 (2018), 18:1–18:18. 2

[BSG12] BARBIČ J., SIN F., GRINSPUN E.: Interactive editing of deformable simulations. *ACM Trans. Graph. 31*, 4 (2012), 70:1–70:8. 11

[BZSL14] BRUNA J., ZAREMBA W., SZLAM A., LECUN Y.: Spectral networks and locally connected networks on graphs. *International Conference on Learning Representations* (2014). 2

[CAJ09] CHADWICK J. N., AN S. S., JAMES D. L.: Harmonic shells: a practical nonlinear sound model for near-rigid thin shells. *ACM Trans. Graph. 28*, 5 (2009), 119:1–119:10. 11

[CCS12] CORSINI M., CIGNONI P., SCOPIGNO R.: Efficient and flexible sampling with blue noise properties of triangular meshes. *IEEE Transactions on Visualization and Computer Graphics 18*, 6 (2012), 914–924. 4

[CdGDS13] CRANE K., DE GOES F., DESBRUN M., SCHRÖDER P.: Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 courses* (2013), SIGGRAPH '13. 3

[CHK13] CAMPEN M., HEISTERMANN M., KOBBELT L.: Practical anisotropic geodesy. *Computer Graphics Forum 32*, 5 (2013), 63–71. 5

[CLB*09] CHUANG M., LUO L., BROWN B. J., RUSINKIEWICZ S., KAZHDAN M.: Estimating the Laplace–Beltrami operator by restricting 3d functions. *Computer Graphics Forum 28*, 5 (2009), 1475–1484. 9

[CLR90] CORMEN T. H., LEISERSON C. E., RIVEST R. L.: *Introduction to Algorithms*. MIT Press, 1990. 5

[CPK17] CHOUKROUN Y., PAI G., KIMMEL R.: Schrödinger Operator for Sparse Approximation of 3D Meshes. In *Symposium on Geometry Processing 2017-Posters* (2017). doi:10.2312/sgp.20171205. 2

[CSBK16] CHOUKROUN Y., SHTERN A., BRONSTEIN A., KIMMEL R.: Elliptic operator for shape analysis. *ArXiv abs/1611.01990* (2016). 2, 11

[DBG*06] DONG S., BREMER P.-T., GARLAND M., PASCUCCI V., HART J. C.: Spectral surface quadrangulation. *ACM Trans. Graph. 25*, 3 (2006), 1057–1066. 2

[DBV16] DEFFERRARD M., BRESSON X., VANDERGHEYNST P.: Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems* (2016), pp. 3844–3852. 2

[DM05] DRINEAS P., MAHONEY M. W.: On the nyström method for approximating a gram matrix for improved kernel-based learning. *J. Mach. Learn. Res. 6* (2005), 2153–2175. 2

[FBCM04] FOWLKES C., BELONGIE S., CHUNG F., MALIK J.: Spectral grouping using the nystrom method. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, 2 (2004), 214–225. 2

[GBAL09] GEBAL K., BÆRENTZEN J. A., AANÆS H., LARSEN R.: Shape analysis using the auto diffusion function. *Computer Graphics Forum 28*, 5 (2009), 1405–1413. 2

[GJ*10] GUENNEBAUD G., JACOB B., ET AL.: Eigen v3. http://eigen.tuxfamily.org, 2010. 7

[HMT11] HALKO N., MARTINSSON P. G., TROPP J. A.: Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review 53*, 2 (2011), 217–288. 3

[HSvTP10] HILDEBRANDT K., SCHULZ C., VON TYCOWICZ C., POLTHIER K.: Eigenmodes of surface energies for shape analysis. In *Advances in Geometric Modeling and Processing* (2010), vol. 6130 of *Lecture Notes in Computer Science*, Springer, pp. 296–314. 2, 11

[HSvTP11] HILDEBRANDT K., SCHULZ C., VON TYCOWICZ C., POLTHIER K.: Interactive surface modeling using modal analysis. *ACM Trans. Graph. 30*, 5 (2011), 119:1–119:11. 11

[HSvTP12a] HILDEBRANDT K., SCHULZ C., VON TYCOWICZ C., POLTHIER K.: Interactive spacetime control of deformable objects. *ACM Trans. Graph. 31*, 4 (2012), 71:1–71:8. 11

[HSvTP12b] HILDEBRANDT K., SCHULZ C., VON TYCOWICZ C., POLTHIER K.: Modal shape analysis beyond Laplacian. *Computer Aided Geometric Design 29*, 5 (2012), 204–218. 2

[HWAG09] HUANG Q., WICKE M., ADAMS B., GUIBAS L.: Shape decomposition using modal analysis. *Computer Graphics Forum 28*, 2 (2009), 407–416. 2

[HZM*08] HUANG J., ZHANG M., MA J., LIU X., KOBBELT L., BAO H.: Spectral quadrangulation with orientation and alignment control. *ACM Trans. Graph. 27*, 5 (2008), 1–9. 2

[JBPS11] JACOBSON A., BARAN I., POPOVIC J., SORKINE O.: Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph. 30*, 4 (2011), 78:1–78:8. 4

[JP*16] JACOBSON A., PANOZZO D., ET AL.: libigl: A simple C++ geometry processing library, 2016. http://libigl.github.io/libigl/. 7

[KBB*13] KOVNATSKY A., BRONSTEIN M. M., BRONSTEIN A. M., GLASHOFF K., KIMMEL R.: Coupled quasi-harmonic bases. *Comput. Graph. Forum 32*, 2 (2013), 439–448. 2

[KG00] KARNI Z., GOTSMAN C.: Spectral compression of mesh geometry. In *ACM SIGGRAPH* (2000), pp. 279–286. 2

[LHJ*14] LING R., HUANG J., JÜTTLER B., SUN F., BAO H., WANG W.: Spectral quadrangulation with feature curve alignment and element size control. *ACM Trans. Graph. 34*, 1 (2014), 11:1–11:11. 2

[LJC17] LIU D., JACOBSON A., CRANE K.: A dirac operator for extrinsic shape analysis. *Computer Graphics Forum 36*, 5 (2017). 2, 11

[LMBB17] LEVIE R., MONTI F., BRESSON X., BRONSTEIN M. M.: Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *ArXiv abs/1705.07664* (2017). 2

[LRBB17] LITANY O., RODOLÀ E., BRONSTEIN A. M., BRONSTEIN M. M.: Fully spectral partial shape matching. *Comput. Graph. Forum 36*, 2 (2017), 247–258. 2

[MAB*15] MUSIALSKI P., AUZINGER T., BIRSAK M., WIMMER M., KOBBELT L.: Reduced-order shape optimization using offset surfaces. *ACM Trans. Graph. 34*, 4 (2015), 102:1–102:9. 2

[MRCB18] MELZI S., RODOLÀ E., CASTELLANI U., BRONSTEIN M. M.: Localized manifold harmonics for spectral shape analysis. *Computer Graphics Forum 37* (2018). 2

[NLCK05] NADLER B., LAFON S., COIFMAN R. R., KEVREKIDIS I. G.: Diffusion maps, spectral clustering and eigenfunctions of Fokker–Planck operators. In *Proceedings of the 18th International Conference on Neural Information Processing Systems* (2005), pp. 955–962. 2, 10

[NVT*14] NEUMANN T., VARANASI K., THEOBALT C., MAGNOR M., WACKER M.: Compressed manifold modes for mesh processing. *Comput. Graph. Forum 33*, 5 (2014), 35–44. 12

[OBCS*12] OVSJANIKOV M., BEN-CHEN M., SOLOMON J., BUTSCHER A., GUIBAS L.: Functional maps: A flexible representation of maps between shapes. *ACM Trans. Graph. 31*, 4 (2012), 30:1–30:11. 2

[OCB*16] OVSJANIKOV M., CORMAN E., BRONSTEIN M., RODOLÀ E., BEN-CHEN M., GUIBAS L., CHAZAL F., BRONSTEIN A.: Computing and processing correspondences with functional maps. In *SIGGRAPH ASIA 2016 Courses* (2016), ACM, pp. 9:1–9:60. 2

[Pat17] PATANÈ G.: Accurate and efficient computation of laplacian spectral distances and kernels. *Comput. Graph. Forum 36*, 1 (2017), 184–196. 3

[RCB*17] RODOLÀ E., COSMO L., BRONSTEIN M. M., TORSELLO A., CREMERS D.: Partial functional correspondence. *Comput. Graph. Forum 36*, 1 (2017), 222–236. 2

[ROA*13] RUSTAMOV R. M., OVSJANIKOV M., AZENCOT O., BEN-CHEN M., CHAZAL F., GUIBAS L.: Map-based exploration of intrinsic shape differences and variability. *ACM Trans. Graph. 32*, 4 (2013), 72:1–72:12. 2

[Rus07] RUSTAMOV R. M.: Laplace–Beltrami eigenfunctions for deformation invariant shape representation. In *Symposium on Geometry Processing* (2007), pp. 225–233. 2

[RWP05] REUTER M., WOLTER F.-E., PEINECKE N.: Laplace-spectra as fingerprints for shape matching. In *Proceedings of the ACM Symposium on Solid and Physical Modeling* (2005), pp. 101–106. 2, 3

[RWP06] REUTER M., WOLTER F.-E., PEINECKE N.: Laplace-Beltrami spectra as "Shape-DNA" of surfaces and solids. *Computer-Aided Design 38*, 4 (2006), 342–366. 2, 3, 10

[Sch13] SCHULZ C.: *Interactive Spacetime Control of Deformable Objects and Modal Shape Analysis beyond Laplacian*. PhD thesis, Freie Universität Berlin, 2013. URL: http://www.diss.fu-berlin.de/diss/receive/FUDISS_thesis_000000095730. 11

[SHKvL09] SHARMA A., HORAUD R. P., KNOSSOW D., VON LAVANTE E.: Mesh segmentation using Laplacian eigenvectors and Gaussian mixtures. In *Manifold Learning and Its Applications* (2009). 2

[SLMR14] SONG R., LIU Y., MARTIN R. R., ROSIN P. L.: Mesh saliency via spectral processing. *ACM Trans. Graph. 33*, 1 (2014), 6:1–6:17. 2

[SOG09] SUN J., OVSJANIKOV M., GUIBAS L. J.: A concise and provably informative multi-scale signature based on heat diffusion. *Computer Graphics Forum 28*, 5 (2009), 1383–1392. 2

[SvWW00] SMOLYANOV O. G., VON WEIZÄCKER H., WITTICH O.: Brownian motion on a manifold as limit of stepwise conditioned standard brownian motions. In *Stochastic processes, physics and geometry: new interplays, II (Leipzig, 1999)*. CMS, 2000. 5

[TKMR13] TALWALKAR A., KUMAR S., MOHRI M., ROWLEY H.: Large-scale svd and manifold learning. *Journal of Machine Learning Research 14* (2013), 3129–3152. 2

[VBCG10] VAXMAN A., BEN-CHEN M., GOTSMAN C.: A multi-resolution approach to heat kernels on discrete surfaces. *ACM Trans. Graph. 29*, 4 (2010), 121:1–121:10. 3

[VL08] VALLET B., LÉVY B.: Spectral geometry processing with manifold harmonics. *Computer Graphics Forum 27*, 2 (2008), 251–260. 2, 3, 9, 10

[VMHB14] VÁŠA L., MARRAS S., HORMANN K., BRUNNETT G.: Compressing dynamic meshes with geometric Laplacians. *Computer Graphics Forum 33*, 2 (2014), 145–154. 2

[vTSSH13] VON TYCOWICZ C., SCHULZ C., SEIDEL H.-P., HILDEBRANDT K.: An efficient construction of reduced deformable objects. *ACM Trans. Graph. 32*, 6 (2013), 213:1–213:10. 11

[WBH*07] WARDETZKY M., BERGOU M., HARMON D., ZORIN D., GRINSPUN E.: Discrete quadratic curvature energies. *Computer Aided Geometric Design 24*, 8-9 (2007), 499–518. 3

[WBPS17] WANG Y., BEN-CHEN M., POLTEROVICH I., SOLOMON J.: Steklov geometry processing: An extrinsic approach to spectral shape analysis. *ArXiv abs/1707.07070* (2017). 2, 11

[WJBK15] WANG Y., JACOBSON A., BARBIČ J., KAVAN L.: Linear subspace design for real-time shape deformation. *ACM Trans. Graph. 34*, 4 (2015), 57:1–57:11. 4

[WS01] WILLIAMS C. K. I., SEEGER M.: Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*. MIT Press, 2001, pp. 682–688. 2

[YLX*15] YANG Y., LI D., XU W., TIAN Y., ZHENG C.: Expediting precomputation for reduced deformable simulation. *ACM Trans. Graph. 34*, 6 (2015), 243:1–243:13. 11

[ZvKD10] ZHANG H., VAN KAICK O., DYER R.: Spectral mesh processing. *Computer Graphics Forum 29*, 6 (2010), 1865–1894. 2